150 ptas.



CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IV - Fascículo 40

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-005-8 (tomo 4)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 178410
Impreso en España - Printed in Spain - Octubre 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Ajedrez por ordenador

Analizamos aquí algunas de las ideas en que se basa el desarrollo de programas de este apasionante y milenario juego



Campeón de ajedrez
David Levy es un maestro internacional de ajedrez que abandonó la competición contra seres humanos en 1978. En 1968 aportó una gran suma de dinero, diciendo que ningún programa de ajedrez por ordenador sería capaz de ganarle a él durante los diez años siguientes. Desde entonces, el período cubierto por la apuesta se ha ampliado, pero él continúa invicto. Levy, una autoridad de primera línea en ajedrez por ordenador, dirige Intelligent Software, una empresa que proporciona las estructuras de programación exclusivas sobre las que se basan muchos ordenadores de ajedrez y paquetes de ajedrez por ordenador. Levy piensa que actualmente los micros están empezando a acercarse al rendimiento de los grandes ordenadores en cuanto a jugar al ajedrez, y calcula que dentro de cinco a ocho años un microordenador podrá derrotar a Belle (una máquina para jugar al ajedrez exclusivamente) y a Cray Blitz (un ordenador central que en 1983 ganó el Campeonato Mundial de Ajedrez por Ordenador), posiblemente mediante la utilización de microprocesadores paralelos

Pocos juegos han captado jamás la imaginación tanto como el ajedrez: este juego lo han practicado millones de personas en todo el mundo durante miles de años, y en la actualidad se juega con reglas que prácticamente no han sufrido ninguna modificación desde el siglo XVII. Hay quienes dedican toda su vida al estudio y al dominio de este juego de estrategia, encontrando satisfacción en su exigencia de rigor y agilidad intelectual. El juego ha generado una gama de variantes que intentan introducir mayores niveles de complejidad: por ejemplo, el ajedrez tridimensional implica varios tableros suspendidos en el espacio y exige muchísima más concentración. Otra variante es el ajedrez para tres personas, que se juega en un tablero en forma de Y. En las diagonales donde se intersectan las tres "alas", al movimiento de las piezas se le aplican reglas especiales. La teoría sobre la que se apoya esta versión es que dos de los jugadores se unan contra el tercero y luego luchen entre sí por la victoria. Pero ninguna de estas variantes ha logrado desplazar a la confrontación esencial entre dos personas, una de ellas con las fichas blancas y la otra con las negras, que se enfrentan sobre el tablero dividido en 64 casillas.

Uno de los motivos que inciden en esto es el número casi infinito de variantes dentro del propio juego. En 1949, el matemático Claude Shannon escribió un ensayo titulado *Programming a computer for playing chess* (Programación de un ordenador para jugar al ajedrez), en el que calculaba que hay 10^{120} partidas posibles de 40 movimientos. Esto significa que una persona que juegue al ajedrez las 24 horas del día, siete días a la semana, a razón de una hora con cada partida (que no es mucho tiempo para 40 movimientos) ¡tardaría un poco más de 10^{17} años en efectuar todas las partidas posibles! Por supuesto, en la actualidad el ajedrez se ha analizado tan profundamente que esta vasta gama de posibilidades en la práctica disminuye, según un factor que depende de la experiencia que tenga el jugador.

Dada esta complejidad, no es sorprendente que la programación de ordenadores para jugar al ajedrez haya consumido tanto tiempo y tanto esfuerzo. Los programas de ajedrez se han ejecutado en grandes ordenadores durante muchos años, y en la actualidad existen numerosas versiones para microordenadores personales. El desarrollo de programas de ajedrez de gran calidad para microordenadores está relacionado con las innovaciones en materia de hardware; los elementos problemáticos siempre han sido la falta de memoria suficiente y la relativamente baja velocidad de proceso, pero los avances que se han ido produciendo durante estos últimos años en la tecnología han significado que la calidad de dichos programas depende ahora del software.

Dado que los ordenadores son, esencialmente, calculadoras de gran velocidad, el ajedrez por orde-



Estrategias de software

En un esfuerzo por evaluar algunos de los programas de ajedrez más populares para micros personales, esta obra organizó un minitorneo entre estos productos: Sargon III, participando en un Apple IIe (disco de Hayden Software, programación de Dan y Kathe Spracklen); Cyrus IS Chess, en un Spectrum de 48 K (cassette de Sinclair Software, programación de Intelligent Software); Colossus 2.0, en un Commodore 64 (disco de CDS MicroSystems, programación de Martin Bryant); y Grand Master 64, también para el Commodore 64 (cassette de Audiogenic, programación de Kingsoft).
Aunque estos programas ya se habían enfrentado entre sí en torneos internacionales de ajedrez por microordenador, nosotros deseábamos una evaluación informal basada en características, manejabilidad y competencia. El minitorneo consistió en un mínimo de dos partidas para cada programa, la primera en el nivel de juego más simple y la segunda en el nivel de competición. No hubo ningún intento por designar un ganador general.

Uno de los problemas que surge cuando juegan un ordenador de ajedrez contra otro es que suele ser difícil identificar el nivel de juego que les proporciona a los dos programas "inteligencia" similar. Los niveles se suelen definir por el tiempo que el ordenador se concede a sí mismo para el mejor movimiento, pero podría no existir una correlación directa entre un lapso de 10 segundos en un programa y el mismo límite de tiempo en otro. Por ejemplo, Sargon III le "roba" tiempo a su oponente y mantiene su generador de movimientos en marcha mientras está moviendo su oponente. Todos los otros programas apagan sus generadores de movimientos en este punto. No obstante, se hizo lo posible para ser imparciales, si no absolutamente precisos, al emparejar los programas.

Calidad de juego

En general, todos los programas jugaban un ajedrez sólido, aunque sin inspiración, en el nivel inferior (tardando aproximadamente 10 segundos por movimiento). Y todos hacían algunos movimientos muy extraños, inútiles en apariencia, hacia el final de la fase intermedia de la partida. Probablemente esto fuera consecuencia de una posición "tranquila" en la que los ordenadores tan sólo consumían su tiempo hasta que sucediera algo interesante. En el nivel superior, de competición (alrededor de 10 minutos por movimiento), los cuatro programas mostraron una jugada táctica inteligente y en ocasiones brillante. El cuadro de la página contigua refleja los resultados del torneo.

nador se diseña basándose en cálculos numéricos, que se utilizan para evaluar los dos elementos esenciales del juego: el material y la movilidad. El "material" de un juego se refiere al número y la fuerza de las piezas del tablero. El programa de ajedrez le asigna a cada pieza un valor numérico. Al rey se le puede dar o bien un valor infinito o uno arbitrariamente alto, como 10 000 (esto se hace porque la pérdida del rey da el juego por terminado); a la reina se le asigna un valor de nueve; la torre vale cinco; los alfiles y los caballos valen tres, y los peones, uno. Al considerar si vale la pena sacrificar una pieza con el objeto de capturar una de las piezas del contrincante, el programa compara sus valores. La mayoría de los programas de ajedrez por ordenador conceden una gran importancia a los valores relativos y raramente cambian piezas si ello produce una desventaja material, a menos que se obtenga una notable mejora en cuanto a la fuerza posicional.

La "movilidad" reviste una gran importancia en el ajedrez, ya que todas las piezas tienen poco valor si su movimiento está restringido. Por el contrario, su valor aumenta si se la puede colocar de modo que ejerza influencia sobre varios cuadros a la vez. El programa de ajedrez necesita, en consecuencia, evaluar la movilidad así como las consideraciones relativas al material. Además, el programa debe ser capaz de planear con anticipación, determinando la mejor secuencia de movimientos a partir de cualquier posición dada. Es aquí donde los programas de ajedrez pueden demostrar su superioridad, utilizando la velocidad del ordenador para examinar un gran número de posibles jugadas en un lapso muy corto.

La mayoría de los programas de ajedrez utilizan una técnica de "fuerza bruta", buscando tantos movimientos como sea posible en el tiempo permitido. El tiempo destinado para cada movimiento se determina seleccionando un "nivel" de juego al principio de cada partida; cada nivel da una medida de tiempo diferente durante la cual el ordenador debe realizar un movimiento. Estos períodos varían desde algunos segundos hasta varias horas, y cuanto más tiempo se le conceda al ordenador para pensar, mayores son las probabilidades de que encuentre la mejor línea de ataque para la posición en la que se encuentra.

En cada movimiento el ordenador determina si el rey está o no en jaque, y luego investiga si hay piezas amenazadas en cualquier bando, si se pueden ocupar posiciones clave y muchas otras consideraciones similares. Cuantos más criterios examine el ordenador, mejor será el resultado. La cuestión final consiste en descubrir si se puede forzar al rey del contrincante a una posición de jaque mate.

En los juegos entre ordenadores y seres humanos, los primeros tienen una clara ventaja en cuanto a velocidad y alcance de búsqueda; aun así, un jugador humano excelente siempre derrotará a un programa para ordenador excelente, debido a la capacidad de los humanos para ver y crear nuevas aperturas y posiciones. Los ordenadores practican un ajedrez táctico soberbio, pero, incluso entre los



Cyrus IS Chess hizo tablas con Colossus y venció a Grand Master en el nivel simple, e hizo tablas con Sargon III en el nivel de competición.

Colossus hizo tablas con Cyrus IS Chess en el nivel simple, venció a Grand Master en el nivel de competición e hizo tablas con Sargon III en el nivel de competición.

Sargon III perdió con Grand Master en el nivel simple e hizo tablas con Cyrus y Colossus en el nivel de competición.

Grand Master venció a Sargon III y perdió con Cyrus IS Chess en el nivel simple, y perdió con Colossus en el nivel de competición.

Características

Todos los programas de ajedrez competentes deben incluir la capacidad de enrocar, promocionar un peón en una reina y capturar al paso, y determinarán situaciones de tablas por ahogado. Algunos de estos programas poseen interesantes características adicionales. Sargon III es el programa que tiene más extras e incluye un segundo disco que contiene 107 partidas de ajedrez clásicas y 45 problemas de ajedrez. La documentación es excelente, con 75 páginas en un cuaderno de hojas sueltas. Por supuesto, Sargon III se ejecutó en un Apple IIe y salió tres veces más caro que los otros programas. Por precio, Cyrus IS Chess y Colossus también ofrecen algunas atractivas características, como se puede apreciar en la tabla.

Características	Grand Master 64	Colossus	Cyrus IS	Sargon III
Jugada invisible	NO	SI	NO	NO
Lista movimientos posibles	NO	SI	NO	NO
Juega "la siguiente mejor"	NO	NO	SI	NO
Vuelve a jugar una partida	NO	SI	SI	SI
Muestra listado	NO	SI	NO	SI
Imprime listado	NO	NO	SI	SI
Retira movimientos	SI	SI	SI	SI
Enseñanza	SI	NO	NO	SI
Promociona a no reina	NO	SI	SI	SI
Partida entre humanos	NO	SI	SI	SI
Cambia de lado	SI	SI	SI	SI
Analiza problemas	NO	SI	SI	SI
Muestra búsqueda	SOLO 1 JUGADA	SI	NO	SI
Invertir tablero	SI	SI	SI	SI
Tablas ofrecidas	NO	NO	NO	SI
Imprime tablero	NO	NO	SI	SI
Guarda partida	NO	SI	SI	SI
Inhabilita librería	NO	NO	NO	SI
Partida automática	NO	SI	SI	SI
Reloj de tiempo real	SI	SI	NO	SI

Conclusión

En términos de manejabilidad, Cyrus IS Chess y Colossus son los más fáciles de utilizar porque los movimientos son entrados empleando el cursor, mientras que Sargon III y Grand Master exigen que se entren los movimientos en notación algebraica, como E2-E4 para peón 4 rey. Colossus y Sargon poseen las mejores visualizaciones en pantalla.

maestros humanos de ajedrez, un buen jugador posicional siempre le ganará a un buen jugador táctico. Los programadores de ajedrez por ordenador se han centrado en la táctica porque, para un ordenador, el juego táctico implica un proceso numérico muy simple. Si un contrincante humano realiza un movimiento poco convencional, a menudo el ordenador no ofrecerá la mejor respuesta. Por este motivo, muchos programas de ajedrez tienen dificultades para tratar con las posiciones "tranquilas", en las que ninguno de los movimientos disponibles ofrece una determinada ventaja táctica. En estas situaciones, que se producen con frecuencia en la fase final de la partida, el programa a menudo revolverá las piezas por ahí en vez de aprovechar la oportunidad que se le brinda para planear sus futuros movimientos.

Un estilo de programación que se ha desarrollado recientemente implica la "búsqueda selectiva". Utilizando esta técnica, el ordenador imita a un jugador humano, analizando con mayor profundidad un menor número de posibles movimientos. Hegener y Glaser, en Alemania, han utilizado la técnica de búsqueda selectiva en su programa *Mephisto III*, que busca todos los movimientos posibles para los dos primeros movimientos de un jugador, luego va estrechando la búsqueda y examina una gama de movimientos más pequeña pero en mayor profundidad. El *Mephisto III* también intenta distinguir entre posiciones tranquilas y posiciones tácticas. Las técnicas de este tipo convertirán a los ordenadores en un auténtico desafío para los jugadores humanos.

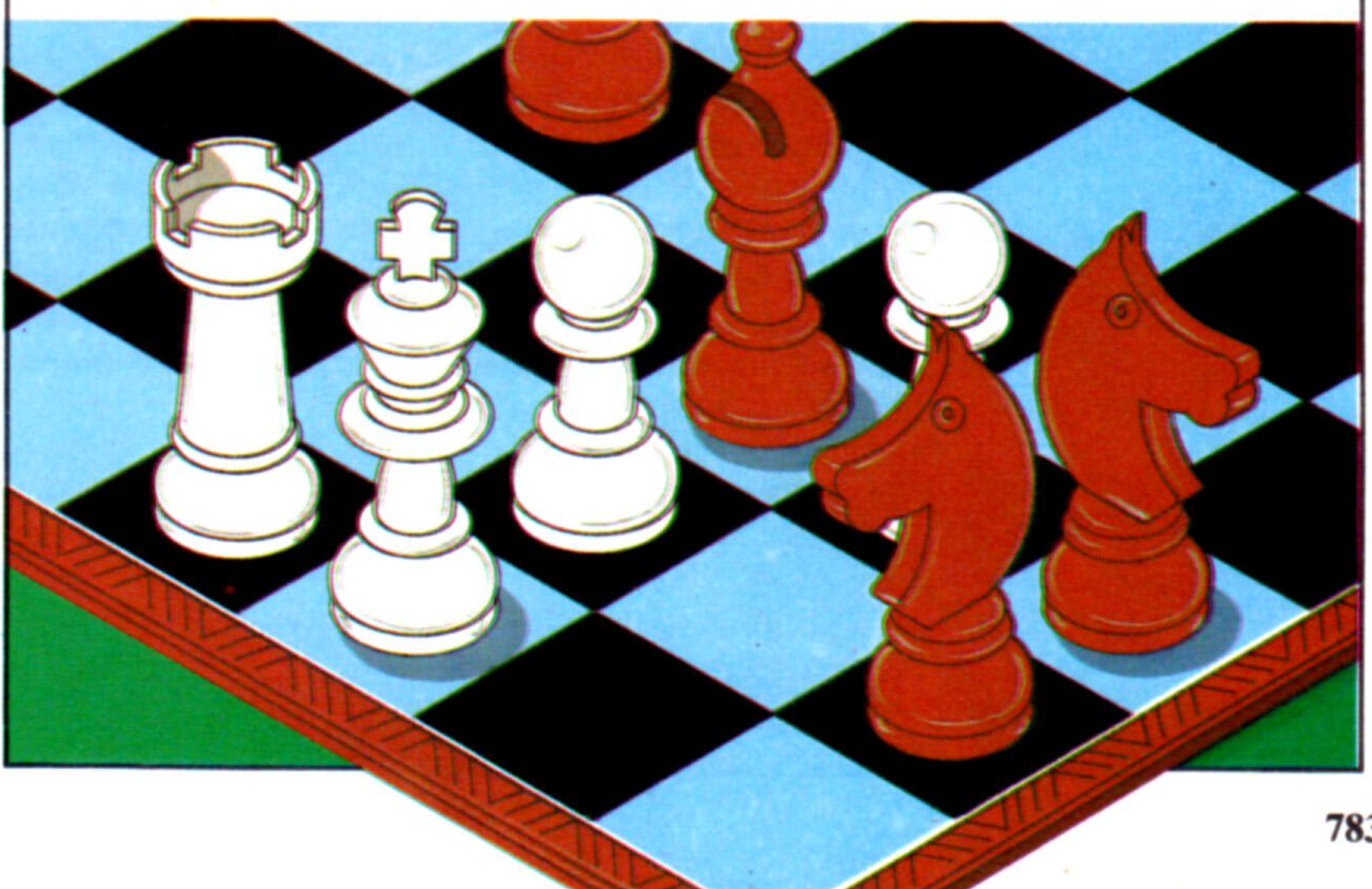
Corazón contra cabeza

La capacidad para examinar todos los movimientos hasta nueve niveles por adelantado casi garantiza a los programas de ajedrez una superioridad táctica respecto a los seres humanos. La destreza especial de un maestro de ajedrez humano reside en seleccionar unos pocos movimientos cruciales en los que concentrar una enorme destreza en el análisis de hasta los próximos 30 movimientos.
Aquí, Moritz (negras) juega con Emmerich (blancas) en 1922. Las negras pueden dar mate sacrificando su reina y efectuando luego tres movimientos muy elegantes con el caballo; la mayoría de los jugadores de ajedrez preferirían esta secuencia a todas las demás. Moritz no lo comprendió así y lamentó su error. Todos nuestros paquetes descubrieron el mate, pero ninguno sugirió la secuencia de movimientos del caballo, aunque algunos la debieron considerar. Esta incapacidad del ordenador de percibir ese final como el "mejor" parece ofrecer a los humanos la única defensa posible contra el ajedrez de la máquina



Movimientos del caballo:

1		H5-H2
2	G1-H2	E5-G4
3	H2-G1	F4-H3
4	G1-F1	G4-H2 mate



Roy Ingram

Punto por punto

Impresión artística
Estos listados ilustran el tipo de gráficos que se pueden realizar con ciertas impresoras matriciales. Cada aguja del cabezal de impresión se controla de forma individual, y se pueden producir algunos patrones complejos y muy satisfactorios. En futuros capítulos del curso se proporcionarán detalles sobre cómo hacerlo. Estas imágenes se crearon utilizando Paintbox, de Print'n'Plotter Products

Una impresora simplifica la labor del programador y resulta imprescindible para el tratamiento de textos

Es probable que el usuario primerizo de un ordenador se quede asombrado ante la gran cantidad de impresoras disponibles en el mercado, dado que existen casi tantas máquinas diferentes como marcas de ordenadores personales. La primera decisión a tomar es determinar el tipo de impresora requerido; es probable que uno se incline por un modelo matricial o de rueda margarita, si bien existen otras variantes, como las impresoras térmicas o las de chorro de tinta. Un modelo margarita produce los resultados de mejor calidad (por lo general, a un precio proporcionalmente elevado) y, por lo tanto, es mejor para el tratamiento de textos; una impresora matricial suele ser más barata, más rápida e ideal para listados y tareas generales de programación. En este capítulo nos concentraremos en las impresoras matriciales.

La velocidad de impresión y la calidad del texto producido por una impresora matricial son puntos importantes a tener en cuenta; los modelos más caros poseen características adicionales, como espaciado proporcional (es decir, a los caracteres más estrechos, como la "i", se les asigna menos espacio que a los anchos, como la "m") y diferentes juegos de caracteres.

La velocidad de impresión es importante, ya que el uso de la impresora "paraliza" al ordenador, ya que el texto se almacena en la memoria de éste hasta que la impresora está preparada para imprimirlo. Por consiguiente, mientras se está llevando a cabo la impresión, el ordenador no se puede utilizar para otras tareas. Las velocidades de impresión se miden en función de los "caracteres por segundo" (cps), de modo que mientras que un modelo caro de 200 cps puede tardar un minuto en imprimir el listado de un programa largo, un modelo más económico con una velocidad de 30 cps tardará más de seis minutos en realizar el mismo listado (y durante esos seis minutos el ordenador no se puede emplear en ninguna otra tarea). Este problema se puede superar utilizando un *buffer* de impresión. Éste consiste simplemente en una placa de circuito



impreso que contiene chips de RAM, que se conecta entre la impresora y el ordenador y almacena los datos mientras la impresora trabaja, dejando libre al ordenador para otras operaciones.

No obstante, las velocidades de impresión que anuncian los fabricantes se deben aceptar con reservas. Al igual que en el caso de las cifras que se dan en relación al consumo de combustible de los automóviles, éstas siempre se dan suponiendo condiciones ideales y con frecuencia tienen poco que ver con la realidad. Las velocidades de impresión se calculan para la impresión de una única línea de texto compuesta del mismo carácter. El texto normal, con sus diferentes caracteres, espacios, saltos de línea y retornos de carro, retarda el cabezal de impresión. Por consiguiente, una impresora con una velocidad nominal de 160 cps probablemente dará un promedio de unos 100 cps al imprimir el listado de un programa.

La calidad de los caracteres producidos sobre el papel varía de modo considerable de una impresora a otra, y depende básicamente de cuántas agujas se utilicen en el cabezal de impresión. Los modelos más baratos emplean apenas siete agujas en el cabezal de impresión, mientras que las máquinas más caras pueden tener 16 o más. En la impresora Commodore, que sólo posee siete agujas, los caracteres se producen como una matriz de puntos de siete por seis. La Canon PW1080, sin embargo, utiliza una matriz de 16 por 23. En consecuencia, los puntos individuales no se ven y los caracteres tienen un aspecto "sólido", claramente definido.

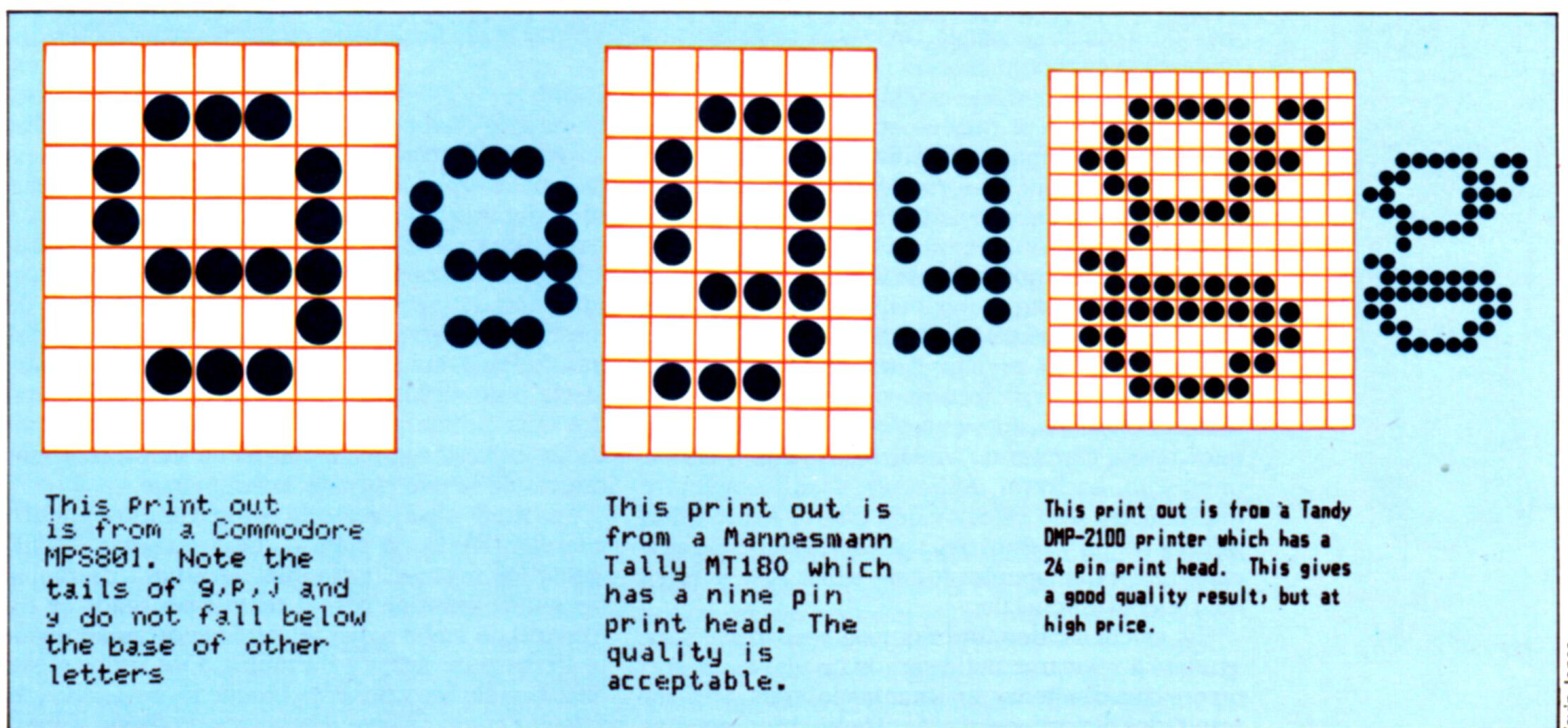
Una impresora matricial es, en realidad, un microordenador dedicado; utiliza chips de memoria ROM y RAM y posee un microprocesador. Como tal, se la puede programar para que haga otras cosas aparte de imprimir texto. Esto se hace enviando códigos de control especiales desde el microordenador a la impresora, o moviendo pequeños interruptores, conocidos como interruptores DIP *(Dual In-line Package)*, situados dentro de la carcasa de la impresora. Por ejemplo, el juego de caracteres ASCII estándar, que está almacenado en la memoria de la impresora, se puede modificar para adaptarlo a distintos alfabetos. En Gran Bretaña, el signo numérico (#) se suele cambiar para que se imprima como el signo de libra esterlina (£).

Otros efectos especiales incluyen caracteres de doble ancho, texto enfatizado (más oscuro, más grueso) y distintos espaciados entre líneas. La Epson FX80 es una de las impresoras matriciales más versátiles y posee más de 70 de estas características de impresión.

La gama de impresoras Epson se ha convertido en algo así como un "estándar industrial". Esto significa que gran parte del software que requiere el uso de una impresora (paquetes de tratamiento de textos, programas de facturación, etc.) asumen que se dispone de una Epson. Éste es un punto importante, ya que el software escrito para una marca puede fácilmente no funcionar en otra.

Hay otras consideraciones que bien podrían influir en la elección de una impresora: la fiabilidad es, por cierto, un factor a tomar en cuenta. Una impresora barata podría estar muy bien para realizar el listado de vez en cuando, pero es poco probable que resista el uso continuado que se le dará a diario a una impresora de oficina. Del mismo modo, el ruido es un factor que no se suele considerar; si a usted le gusta trabajar por las noches, algunas impresoras pueden ser realmente ensordecedoras a la una de la madrugada. ¿Posee una tracción por fricción? Todas las impresoras matriciales se suministran con el mecanismo estándar de tracción por arrastre, que trabaja sólo con papel continuo, ese papel que tiene agujeros para rueda dentada a los lados. Sin embargo, si hay que imprimir hojas de papel indivuales, es necesaria la tracción por fricción.

Por último, el factor que quizá sea el más importante: ¿funcionará con su micro? La mayoría de las impresoras matriciales vienen o bien con un conector Centronics en paralelo o con una interface en serie RS232. Si una impresora no posee la interface adecuada para su micro, en ocasiones se puede instalar una interface alternativa. Aun con la interface adecuada es necesario el cable apropiado para conectar la impresora al ordenador.



Tres distintas impresoras
Estas muestras de impresión ilustran la diferencia entre tres impresoras matriciales. La principal razón de la variación radica en el número de agujas del cabezal de impresión: las que poseen más agujas son las que ofrecen los caracteres más detallados
La muestra de la izquierda corresponde a una Commodore MP8801. Nótese que las "colas" de las letras *g, p, q* e *y* no caen por debajo de la base de las otras letras
En el centro vemos una impresión realizada por una Mannesmann Tally, que cuenta con un cabezal de impresión de nueve agujas. La calidad es aceptable
Por último, a la derecha, podemos apreciar la composición de una impresora Tandy DMP-2100, que posee un cabezal de impresión de 24 agujas y que proporciona un resultado de buena calidad. Su precio es superior al de las dos anteriores impresoras

Un programa versátil

Una hoja electrónica puede servir como "generador de ideas" y constituirse, además, en valiosa ayuda en el tratamiento de la información

Al igual que un procesador de textos o una base de datos, la hoja electrónica posee muchas facilidades que sus usuarios raramente investigan. La mayoría de las personas que disponen de un sistema para tratamiento de textos utilizan pocas veces sus instrucciones más sofisticadas, al igual que los programas de bases de datos tienden a emplearse como sistemas de índice y administración de archivos, desaprovechando sus capacidades para el proceso de datos. No obstante, la mayor parte de los usuarios de ordenadores personales no poseen un programa de hoja electrónica y no son conscientes de la conveniencia de contar con alguno. Muchos piensan que un programa de este tipo les resultará aburrido y de escasa utilidad práctica, y por lo general se sienten intimidados por la asociación de la hoja electrónica con su uso para fines financieros y de gestión. Considerarla desde este punto de vista es menospreciar la importancia de la administración financiera en el hogar y pasar por alto el hecho de que las hojas electrónicas son sencillamente procesadores de ideas que han sido encasillados en la imagen de su uso contable. De hecho, las hojas electrónicas son a los conceptos lo que los procesadores de textos son al texto.

Una hoja electrónica es en realidad un editor de textos y una calculadora, todo en uno. Se la denomina hoja electrónica porque está dividida en filas y columnas, como una hoja de contabilidad, con los datos dispuestos en celdas o casilleros. Éstos se parecen a los casilleros de una hoja de papel en el sentido de que se las puede utilizar de muchas formas diferentes: se puede depositar texto en una celda en la cual permanecerá para su visualización, se pueden entrar datos numéricos para visualización y cálculo, o se pueden entrar fórmulas matemáticas que operen con el contenido de otros casilleros. Una vez que las fórmulas están en su sitio, la hoja electrónica se convierte en un programa generado por el usuario que está a la espera de una entrada. Cada vez que se da entrada a datos nuevos (en forma de texto, datos numéricos o datos algebraicos), todas las celdas de fórmulas se recalculan incorporando los nuevos datos, manteniendo de esta manera la hoja electrónica constantemente actualizada. Ésta puede, por consiguiente, utilizarse para tareas simples de visualización por pantalla-impresora, haciendo que resulte fácil formatear e imprimir no sólo cálculos que podría hacer usted mismo (si no fueran tan tediosos), sino también otros en los que, de otra forma, jamás se le hubiera ocurrido pensar.

En muchos casos, utilizar una hoja electrónica ayudará a revelar necesidades de las que el usuario no era consciente, como llevar inventarios, analizar resultados deportivos, diseñar formularios, generar devoluciones de impuestos, decidir si alquilar o comprar un televisor, etc. Todas estas tareas las podría programar alguien que tuviera conocimientos prácticos de BASIC, pero desarrollar cada una de ellas ocuparía horas, y la mayor parte de este tiempo se perdería elaborando y depurando las infinitas instrucciones PRINT TAB, PRINT AT e INPUT necesarias para formatear la salida por pantalla. La gran ventaja de la hoja electrónica es que uno va formateando la visualización a medida que elabora las relaciones entre las variables. Esto se efectúa con la misma naturalidad con que uno prepararía una hoja de papel, escribiendo el texto, los datos y los resultados de los cálculos en el lugar en donde desearía que se visualizara.

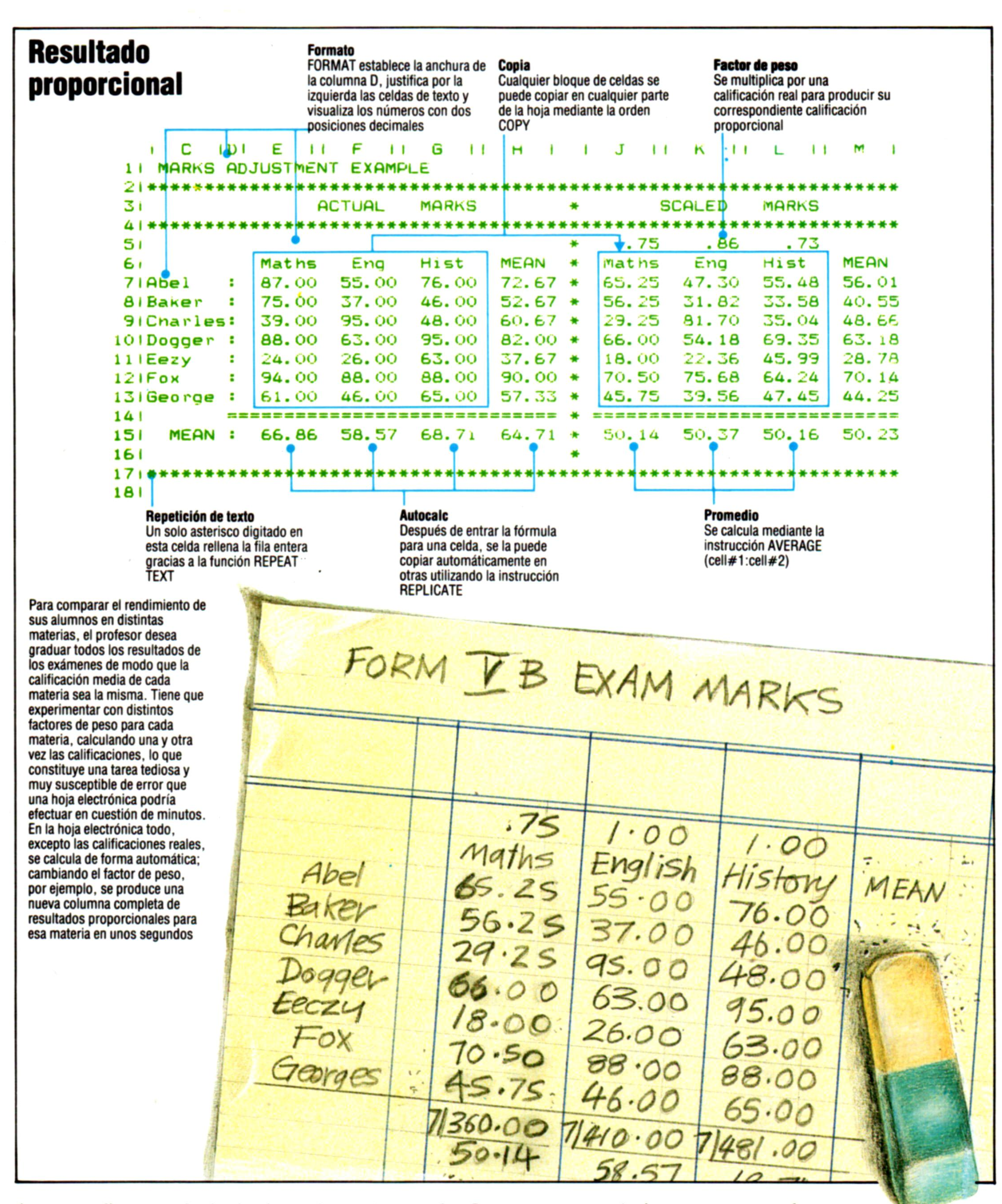
Las hojas electrónicas disponen de diversas instrucciones para simplificar el trazado: uno puede copiar, desplazar o eliminar bloques de celdas, insertar y eliminar filas y columnas y definir el formato de una celda o un bloque en términos de tamaño, justificación (alineación con otros ítems de la misma columna) y posición de la coma decimal. Éstos son precisamente los detalles que son tan difíciles de controlar en la mayoría de las versiones de BASIC, pero que son vitales para el aspecto y la sencillez de uso de cualquier informe.

Las funciones de cálculo son análogas a estas facilidades de formato. Con una única instrucción se puede calcular el valor medio de una fila o columna de datos, contar las entradas de una tabla distintas a cero, elaborar la suma de una matriz de valores, hallar los valores máximo y mínimo de una lista y utilizar estas facilidades en expresiones matemáticas con operadores y funciones más familiares, como "+" y "/", SQR y ABS. No obstante, no todas las hojas electrónicas disponen de estas facilidades.

Quizá la instrucción más práctica de la hoja electrónica sea REPLICATE (repetir). Su utilización permite duplicar un cálculo o valor entrado en una o más celdas, de modo que en una docena de pulsaciones de teclas se pueden obtener tablas de acumulación de datos (como el interés hipotecario de mes a mes, o los gastos domésticos semana a semana). La programación de hojas electrónicas se convierte muy rápidamente en una extensión natural del BASIC aritmético, permitiendo expresar complicadas expresiones matemáticas en una forma más directa de la que permite este lenguaje.

Las hojas electrónicas ya completadas se pueden guardar (SAVE) en cinta o disco y cargar (LOAD) desde los mismos, y muchas versiones ofrecen la opción de guardar sólo el texto y los datos en un formato de archivo que se puede tratar por software de base de datos y tratamiento de textos. Esto hace posible incorporar en bloque los resultados de cálculos en un archivo de textos o de datos, y cons-

Resultado proporcional



	C	D	E	F	G	H	I	J	K	L	M
1	MARKS ADJUSTMENT EXAMPLE										
2	**										
3			ACTUAL	MARKS		*		SCALED	MARKS		
4	**										
5						*	.75	.86	.73		
6			Maths	Eng	Hist	MEAN	*	Maths	Eng	Hist	MEAN
7	Abel	:	87.00	55.00	76.00	72.67 *		65.25	47.30	55.48	56.01
8	Baker	:	75.00	37.00	46.00	52.67 *		56.25	31.82	33.58	40.55
9	Charles	:	39.00	95.00	48.00	60.67 *		29.25	81.70	35.04	48.66
10	Dogger	:	88.00	63.00	95.00	82.00 *		66.00	54.18	69.35	63.18
11	Eezy	:	24.00	26.00	63.00	37.67 *		18.00	22.36	45.99	28.78
12	Fox	:	94.00	88.00	88.00	90.00 *		70.50	75.68	64.24	70.14
13	George	:	61.00	46.00	65.00	57.33 *		45.75	39.56	47.45	44.25
14			==========	==========	==========	========== *		==========	==========	==========	==========
15	MEAN	:	66.86	58.57	68.71	64.71 *		50.14	50.37	50.16	50.23
16						*					
17	**										
18											

Para comparar el rendimiento de sus alumnos en distintas materias, el profesor desea graduar todos los resultados de los exámenes de modo que la calificación media de cada materia sea la misma. Tiene que experimentar con distintos factores de peso para cada materia, calculando una y otra vez las calificaciones, lo que constituye una tarea tediosa y muy susceptible de error que una hoja electrónica podría efectuar en cuestión de minutos. En la hoja electrónica todo, excepto las calificaciones reales, se calcula de forma automática; cambiando el factor de peso, por ejemplo, se produce una nueva columna completa de resultados proporcionales para esa materia en unos segundos

tituye un valioso paso hacia el software integrado. Esto se suele aplicar sólo en los paquetes más caros.

Disponiendo de un razonable conjunto de instrucciones, un programa de hoja electrónica está limitado principalmente por la imaginación del usuario o la cantidad de memoria disponible en el ordenador. Los programas en sí mismos por lo general son extensos, y las aplicaciones con grandes tablas y sofisticadas facilidades para proceso de datos pueden llenar rápidamente el resto de la memoria. Además, los cálculos complicados pueden retardar de forma perceptible la respuesta de cálculo del programa.

Números primos

Veamos a continuación cómo se puede averiguar si un número determinado es primo

En numerosas ocasiones surge la duda de si un determinado número es o no es primo. Entonces se debe recurrir a una serie de operaciones que pueden prolongarse según sea la complejidad de tal número. El ordinograma que ofrecemos en esta página muestra cómo se puede saber con certeza si un número es primo.

El sencillo programa descrito a continuación determina si un número entrado por teclado (contenido en la variable N) es primo o no. Para ello, el programa va dividiendo dicho número por un contador C que empieza con el valor 2 y se va incrementando de uno en uno. Si el resultado de alguna de las sucesivas divisiones es un número entero, significa que el número N es divisible por ese valor de C y, por tanto, no es primo. Si, en cambio, el contador C alcanza el valor de N sin hallar ninguna división con resultado entero, significa que el número es primo.

```
10 REM ********** = NUMEROS PRIMOS
20 INPUT"ENTRE UN NUMERO";N
30 C = 2
40 X = N/C
50 IF X = INT(X) THEN PRINT "EL NUMERO
   ENTRADO NO ES PRIMO" : END
60 C = C +1
70 IF C <> N THEN GOTO 40
80 PRINT "EL NUMERO ENTRADO ES PRIMO"
90 END
```

INICIO
ENTRAR NÚMERO
PONER CONTADOR A DOS
DIVIDIR NÚMERO ENTRE CONTADOR
RESULTADO ENTERO ?
SÍ
NÚMERO NO PRIMO
NO
SUMAR 1 A CONTADOR
CONTADOR = NÚMERO ?
NO
SÍ
NÚMERO PRIMO
FIN



Atractiva solidez

Colour Genie, una máquina grande y sólida, diseñada para uso personal, ofrece una interesante relación posibilidades-precio

Basado en el popular microprocesador Z80, el Colour Genie posee un teclado tipo máquina de escribir con 62 teclas. Éstas incluyen cuatro teclas de función, dos Reset (que se deben pulsar simultáneamente) y una Mode Select (selección de modo), que permite obtener desde el teclado caracteres gráficos predefinidos.

La máquina tiene 32 Kbytes de memoria. De éstos, hay dos Kbytes separados para el sistema, y los gráficos de alta resolución utilizan otros cuatro Kbytes. Los 16 Kbytes de ROM contienen una versión ampliada del BASIC Microsoft, que no ofrece ninguna de las características de programación estructurada de que disponen versiones de BASIC más recientes. Sin embargo, admite variables enteras, doble precisión, matrices multidimensionales de cualquier tipo de variables, y amplias facilidades para manipulación de strings. Incluye muchas y muy prácticas instrucciones para realizar efectos de sonido y gráficos de alta resolución.

Las facilidades de sonido son relativamente sofisticadas: ofrecen tres canales (que permiten tocar cuerdas) y posibilitan la salida a través del televisor. Dos instrucciones de BASIC controlan la generación de sonido: PLAY da un sonido predefinido similar a un repique de campanas, mientras que SOUND permite generar otros ruidos.

A pesar de ser amplias y potentes, las facilidades para gráficos del Colour Genie están actualmente algo anticuadas. La pantalla se considera como dos "páginas" (de hecho, dos zonas distintas de la memoria de pantalla), una de las cuales almacena y visualiza texto, bloques de caracteres para gráficos y caracteres para gráficos predefinidos, mientras que la otra página se utiliza para la visualización de gráficos de alta resolución. En modalidad de textos, el Color Genie puede visualizar hasta 25 líneas de 40 caracteres. En modalidad de gráficos, las dimensiones de visualización son de 160 ×102 pixels, lo que apenas es "alta resolución" para los estándares actuales.

Tratamiento de gráficos

La tecla Mode Select accede a la página de alta resolución, reservando 4 Kbytes de memoria. El BASIC dispone de numerosas instrucciones para tratamiento de gráficos: se pueden trazar líneas, rellenar superficies con bloques de color sólido y definir, dibujar y borrar formas. Cuando se la incorpora en un programa en BASIC, la instrucción FGR visualiza la página para gráficos, aunque al final del programa el ordenador vuelve automáticamente a la modalidad de textos. El BASIC incluye, asimismo, instrucciones para limpiar la página de gráficos (FCLS) y cambiar los colores de fondo (FILL) y primer plano (FCOLOR). Este sistema es más incómodo que la disposición de página única adoptada por



Chris Stevens

El tímido Genie
Un ordenador que nunca ha alcanzado la fama del Spectrum ni del Commodore 64 es el Colour Genie, a pesar de llevar en el mercado aproximadamente el mismo tiempo. Sea como fuere, tiene un grupo de partidarios pequeño pero exclusivo. La máquina tiene 32 Kbytes de memoria y unas palancas de mando bastante inusuales (vienen dos, con teclados numéricos incorporados y con una base propia)



El indicador de cassette del Genie
Intentar adecuar una grabadora de cassette al nivel de volumen correcto para el ordenador puede ser sumamente difícil, pero el Colour Genie posee su propio indicador de cinta para simplificar de manera considerable la escritura y lectura de una cassette

la mayoría de las máquinas nuevas, pero permite colorear cada pixel individualmente (a diferencia del Spectrum, por ejemplo, que posee una resolución más alta pero limita los colores que se pueden visualizar dentro de cada bloque de ocho por ocho pixels). La mayor parte del software de tipo recreativo utiliza la pantalla de textos por la velocidad, con caracteres definidos por el usuario para dar un efecto de alta resolución.

La visualización en pantalla es clara y uniforme, pero el juego de caracteres utilizado hace que resulte algo difícil leer el texto. El Genie dispone de ocho colores (blanco, rojo, verde, amarillo, cyan, magenta, azul y naranja), todos los cuales se pueden visualizar en la pantalla de textos al mismo tiempo. Los gráficos de alta resolución limitan al usuario a cuatro colores (rojo, azul, verde y negro), pero hay una instrucción adicional (BGRD) para poner el fondo de la página de gráficos en rosa.

Se incluyen varias interfaces: una salida RS232 para impresoras y modems; una puerta de ampliación de 50 canales, que se utiliza para conectar unidades de disco; una salida de video compuesta; una salida de audio; un conector para lápiz óptico y una puerta para palanca de mando. Entre los periféricos disponibles se incluyen palancas de mando, una interface para impresora Centronics, un cartucho Prestel (que requiere un modem) y unidades de disco. Las palancas de mando dobles tienen pequeños teclados incorporados pero son difíciles de utilizar (hace falta presionar muy fuerte para que respondan, y las palancas de mando no vuelven a la posición central "neutral" cuando se las suelta). Eaca, la empresa que fabrica el Colour Genie, no ofrece una unidad de disco para la máquina. Hay una disponible gracias a una empresa británica que ofrece su propia unidad de disco utilizando un sistema operativo llamado QDOS, similar al TRS-DOS de Tandy.

La carcasa lleva incorporado un indicador de nivel de grabación para paliar los problemas de carga desde la cassette; el usuario simplemente ajusta el volumen hasta que la aguja se centra, después de lo cual las cintas de cassette se cargan con facilidad. Además, se puede instalar un "estabilizador de datos" entre la grabadora de cassette y el cable para cassette del Genie; éste "limpia" la señal y también ayuda a la operación con las cintas.

El Colour Genie se suministra con dos manuales: una guía para el principiante y un manual básico. Ambos están escritos con claridad pero no son lo suficientemente detallados y ninguno posee índice. En realidad, el manual básico ni siquiera tiene una

Segundos 16 K de memoria
Están en una placa separada porque el Colour Genie originalmente se vendía como una máquina de 16 K con la opción de otros 16 K adicionales. Éstos ahora se incluyen como estándar

Primeros 16 K de memoria
Forman parte de la placa de circuito impreso principal

Indicador de tensión

Salida de video compuesta
Permite utilizar una pantalla

Salida de sonido

Modulador de TV
Produce una señal para televisores normales. Hay un cable que está conectado con él de forma permanente

Interruptor de tensión

Transformador de corriente
Está incorporado en el ordenador

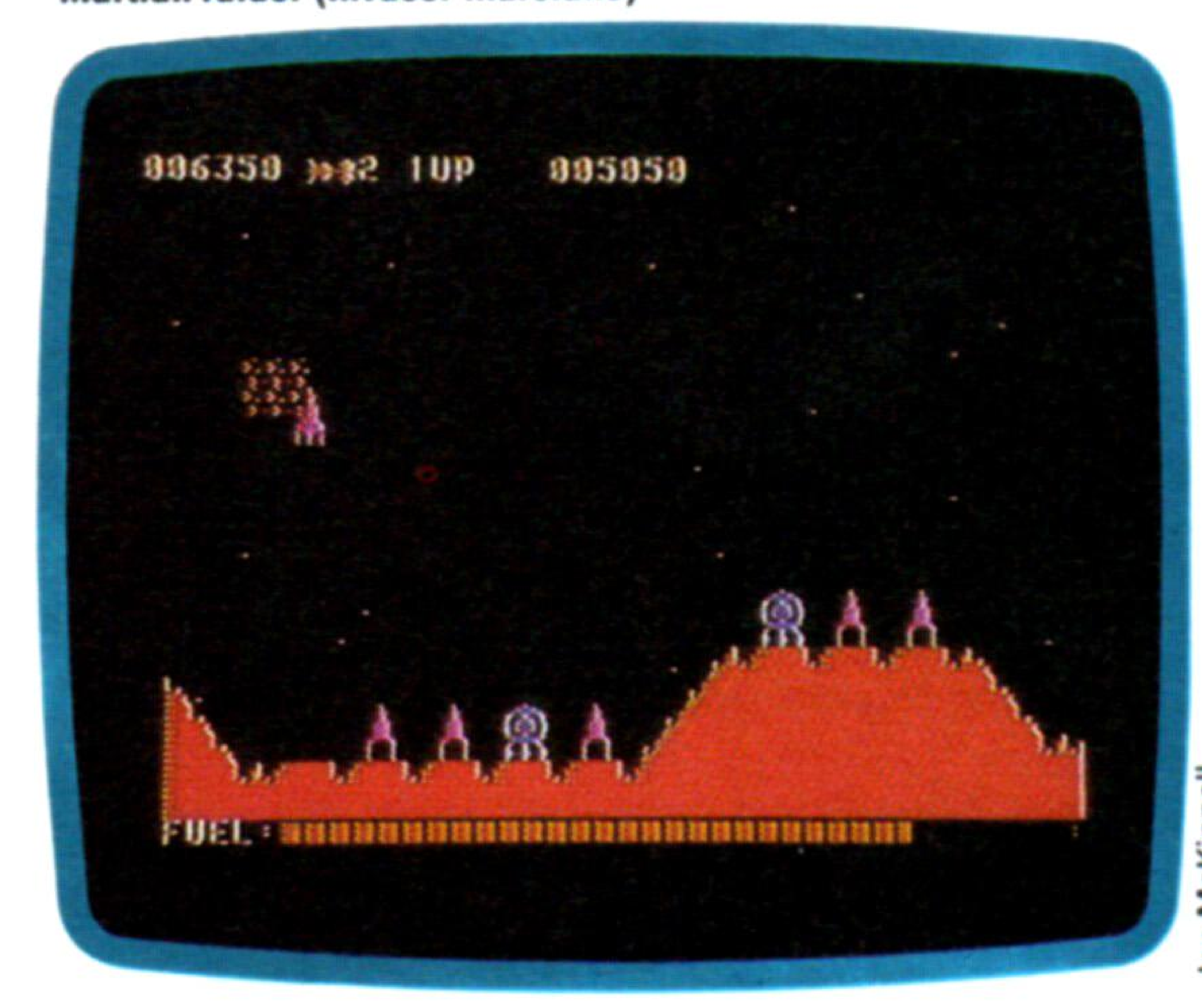
16 K de ROM
La memoria ROM está distribuida en cuatro chips de ROM



página de contenido. Para los usuarios más avanzados existe un manual técnico que se ha de pagar aparte.

A pesar de su aspecto anticuado, el Colour Genie parece ofrecer una buena relación calidad-precio. Cae de lleno en la categoría de "ordenador personal" y tiene poco que ofrecer para el usuario científico o de oficina. La construcción robusta, las buenas capacidades de sonido, la completa gama de periféricos y un BASIC bastante normalizado harán que esta máquina sea especialmente atractiva para el principiante.

Martian raider (Invasor marciano)



Ian McKinnell

Opciones de software
La disponibilidad de software para el Colour Genie es bastante limitada, pero la calidad de lo que se puede obtener por lo general es muy buena. La mayor parte del software son juegos y con frecuencia éstos son versiones traducidas de juegos creados para las máquinas más conocidas

COLOUR GENIE

DIMENSIONES
90 × 280 × 340 mm

CPU
Z80, 2,2 MHz

MEMORIA
32 K de RAM, 16 K de ROM

PANTALLA
Hasta 25 filas de 40 columnas de texto; gráficos hasta 160 × 102 pixels. Ocho colores en modalidad de textos; 4 colores para gráficos

INTERFACES
Palancas de mando (2), puerta RS232, puerta para lápiz óptico, puerta para ampliación

LENGUAJES DISPONIBLES
BASIC incluido

TECLADO
Tipo máquina de escribir, con 62 teclas y cuatro de función

DOCUMENTACION
La máquina se suministra con una guía para el principiante y un manual básico. Ambos son demasiado breves como para ser de mucha utilidad, y ninguno de los dos tiene índice. También hay a la venta un manual técnico

VENTAJAS
El Colour Genie es una buena máquina "familiar". Es de construcción robusta, utiliza BASIC Microsoft, ofrece gráficos razonables y buen sonido, con salida por el televisor

DESVENTAJAS
El diseño del Genie es anticuado; posee un procesador lento y la pantalla se manipula en forma de dos "páginas". Hay poco software para esta máquina

Palancas de mando del Colour Genie
Las palancas de mando del Colour Genie son muy atractivas, pero caras y difíciles de utilizar. Se necesita mucha presión para moverlas y no vuelven a la posición central al soltarlas. Los teclados numéricos incorporados son inusuales y no muy prácticos



Chris Stevens

Arenas peligrosas

Conducir un camión por el desierto no es tarea sencilla cuando no se puede llevar todo el combustible necesario

Nuestro juego se desarrolla en un desierto de 1 000 km de anchura. Aproximadamente cada 100 km hay una posta de aprovisionamiento donde se pueden almacenar tanques de combustible. De vuelta en la base, uno puede hacerse con tantos tanques de gasolina como crea que va a necesitar, cada uno de ellos suficientemente grande como para abastecer al camión de una etapa hasta la siguiente. La travesía a través del desierto sería bastante sencilla si no fuera por un especial detalle: el camión sólo tiene espacio para un máximo de ocho tanques a la vez. Por consiguiente, para conducirlo a través de las arenas es necesario proveerse de combustible en varios puntos de la ruta, yendo y viniendo de un puesto a otro.

Obviamente, el primer objetivo del juego consiste en asegurarse de que no se quedará sin gasolina: el camino hasta la base es muy largo y el desierto no es el lugar más apropiado para realizar una agradable caminata. En segundo lugar, se debe completar la travesía cubriendo la distancia más corta posible y utilizando la menor cantidad posible de tanques. Solucionar esta doble dificultad le resultará bastante sencillo con el programa preparado para ocho tanques.

Sin embargo, podemos modificar el juego para hacer que el problema resulte un poco más acuciante. ¿Qué sucede, por ejemplo, si únicamente se pueden llevar cuatro o seis tanques cada vez? Para investigar estas variantes, sólo hay que alterar el valor de la variable M de la línea 60. Descubrirá que está utilizando la misma técnica pero que se modifican los intervalos entre sus aprovisionamientos de combustible y la cantidad de jornadas. ¿Puede usted elaborar un algoritmo que lo conduzca con seguridad a través del desierto todas las veces? Este algoritmo, una vez encontrado, podría proporcionar la base para establecer un programa que *resuelva* este problema en particular.

Nuestro enigma demuestra una valiosa técnica para resolver problemas en el desarrollo de un programa. Lo primero que debe hacerse es experimentar con la información dada, probar cierta cantidad de ejemplos calculados y luego, si todo va bien, descubrir un patrón común. A partir de éste se puede elaborar un algoritmo, transformándolo luego en un programa. Si se desea desarrollar nuestro juego *Desert trucker* (El camionero del desierto), se pueden agregar gráficos y otros refinamientos, introduciendo dificultades como, por ejemplo, el tener que llevar agua además de gasolina.

Complementos al BASIC

Este programa está escrito en BASIC Microsoft, de modo que se podrá ejecutar en la mayoría de las máquinas que posean una visualización en pantalla de 40 × 25. En el Spectrum se debe insertar LET antes de las sentencias de asignación.

CHR$(26): Sustituir por CLS en el Spectrum, Oric-1, Atmos, Dragon y BBC; y por CHR$(147) en el Commodore 64 y el Vic-20.

MID$(STR$(A(1),2)): Reemplazar por STR$(A(1)) en el Spectrum, y en todas las máquinas en las que PRINT LEN(STR$(2)) dé 1 como resultado.

THEN 1260 & THEN 1300: En el Spectrum, cambiar por THEN GOTO 1260 & THEN GOTO 1300.



```
10 REM ····Camionero del desierto
30 DIM A(10)
40 A(1) = 80: REM Tanques al comienzo
50 S = 1:REM Posición del camion
60 M = 8: REM Max. de tanques en el camion
70 N = 0: REM Total de tanques usados
100 REM Proxima vuelta
110 PRINT CHR$(26): REM Limpiar pantalla
120 PRINT "PUESTO: 1 2 3 4 5 6 7 8 9 10"
130 PRINT "TANQUES:    ";
140 FOR I = 1 TO 10
145 A$ = MID$(STR$(A(I)),2)
147 IF LEN(A$) < 2 THEN LET A$ = "  " + A$:GOTO 147
150 PRINT A$;"   ";:NEXT I:PRINT
160 X = 9 + (S-1)*3:PRINT TAB(X);"-":PRINT TAB(X);"!":PRINT TAB(X-2);"CAMION":PRINT
    TAB(X-1);T
175 IF S = 10 THEN PRINT:PRINT"Lo ha conseguido!!!":GOTO 1400
180 IF T = 0 AND A(S) = 0 THEN PRINT: PRINT "Amigo... sera mejor que empiece a caminar":GOTO
    1400
190 PRINT: PRINT"Tus opciones son:":PRINT
200 IF T > 1 THEN PRINT "A abandonar tanques de combustible"
210 IF A(S) > 0 THEN PRINT "C Coger tanques de combustible"
220 IF T > 0 THEN PRINT "P Un puesto hacia adelante"
230 IF T > 0 AND S > 1 THEN PRINT "R Un puesto hacia atras"
240 PRINT:PRINT "Cual?";:INPUT A$
250 IF T > 1 AND (A$ = "A" OR A$ = "a") THEN 1260
260 IF A(S) > 0 AND (A$ = "C" OR A$ = "c") THEN 1300
270 IF T > 0 AND (A$ = "P" OR A$ = "p") THEN S = S + T = T-1:N = N + 1:GOTO 110
280 IF T >0 AND S > 1 AND (A$ = "R" OR A$ = "r") THEN S = S-1:T = T-1:N = N + 1:GOTO 110
290 GOTO 110
1250 REM Abandonar tanques
1260   PRINT:INPUT "Cuantos?";A
1270 IF A > T OR A <> INT(A) OR A < 0 THEN PRINT:PRINT"Por favor pruebe otra vez":GOTO 1260
1280 A(S) = A(S) + A:T = T-A:GOTO 110
1290 REM Coger tanques
1300 PRINT:INPUT "Cuantos?";A
1310 IF A > A(S) OR A <> INT(A) OR A < 0 THEN PRINT:PRINT "Por favor pruebe otra vez":GOTO 1300
1320 IF A + T > M THEN PRINT: PRINT"Lo siento, solo hay lugar para llevar";M;"tanques":GOTO 1300
1330 T = T + A:A(S) = A(S)-A
1380 GOTO 110
1390 REM Final del juego
1400 PRINT:PRINT "Ha utilizado";N;"tanques de gasolina    ";
1410 PRINT"de los que traia";T
1420 A = 0:FOR I = 2 TO 9:A = A + A(I):NEXT I
1430 PRINT "consigo y ha dejado";A;"en el desierto."
1440 PRINT:PRINT"Entre RUN para volver a jugar"
1450 END
```

Adrian Morgan

Las tres vidas de Willy

"Manic miner" combina hábilmente humor y gráficos originales, lo que lo ha convertido en un programa de entusiasta aceptación

Manic miner, disponible para el ZX Spectrum de 48 Kbytes y el Commodore 64, es fundamentalmente un juego muy sencillo basado en un best-seller anterior llamado *Kong*. El objetivo de este juego era trepar por escaleras y ramas mientras se evitaban obstáculos, en un intento por rescatar a la afligida doncella que el Gran Simio mantenía cautiva. En *Manic miner*, el jugador asume el papel de Miner Willy, prospector del centro minero denominado Surbiton. Willy encuentra el pozo de una mina olvidada del que una civilización perdida extraía oro y otros metales preciosos. Lamentablemente, los antiguos habitantes de la mina se han olvidado de desactivar los Manic Mining Robots (robots mineromaníacos), por lo cual la tarea de recuperar el tesoro, en principio no demasiado ardua, resulta sumamente difícil.

La mina tiene 20 cavernas y cada una de ellas contiene cuatro llaves que hay que coger, para poder abrir la puerta que conduce a la etapa siguiente. Cada caverna tiene diversos salientes sobre los que hay que saltar para poder alcanzar las llaves. Algunos de estos salientes están en malas condiciones (presumiblemente por su antigüedad) y ceden cuando Willy llega a ellos. Las cavernas están rotuladas con frases, como "La guarida de Eugene" (alusión al programador Eugene Evans, joven prodigio de Imagine), "El minero Willy se encuentra con el Rey Bruto", "El ataque de los teléfonos mutantes" (otra "broma para los del ramo", esta vez dirigida al programador Jeff Minter, cuya obsesión son las llamas mutantes) y "Bahía para el aterrizaje del Skylab". Todas las cavernas están habitadas por numerosos seres extraños cuyo solo contacto significa la muerte instantánea. Hasta las plantas son letales.

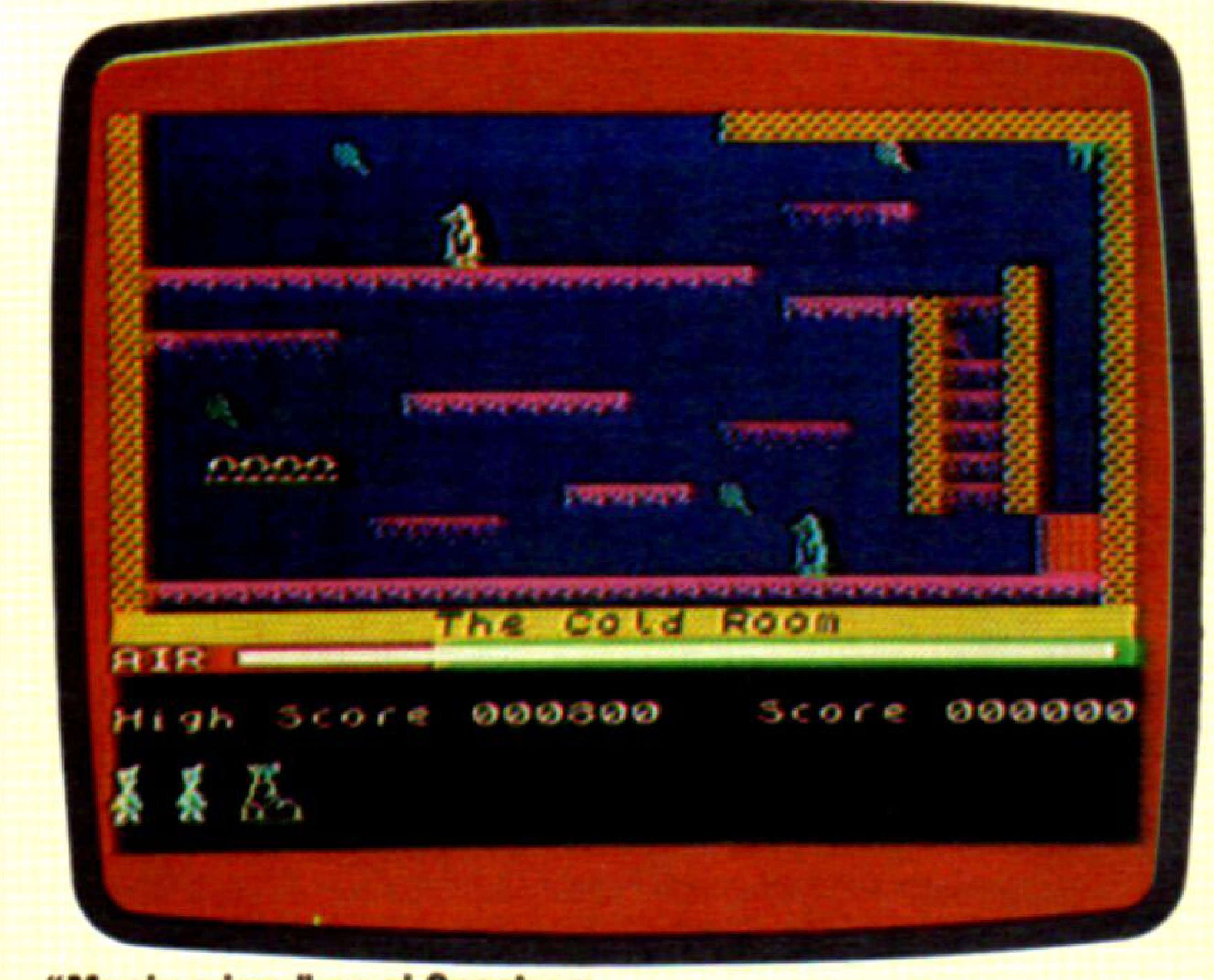
El jugador dirige a Willy para evitar todos estos problemas con estas tres órdenes simples: "Derecha", "Izquierda" y "Salta". Ésta es parte de la atracción del juego: la simplicidad de los mandos significa que no se requiere un largo período de aprendizaje y uno puede seleccionar las teclas con las que se siente más a gusto.

Willy tiene tres vidas y en cada encarnación dispone de una provisión de aire limitada, indicada mediante un medidor que hay en pantalla. Como la pérdida de la tercera vida lleva a Willy de regreso a la Caverna Uno, el juego puede resultar muy frustrante y no es sorprendente que algunos usuarios se las hayan arreglado para modificar el programa con el fin de empezar por la caverna de su elección.

La traducción para el Commodore es una copia casi exacta de la versión para el Spectrum y no consigue sacar partido de las instrucciones para sonido del 64, más versátiles, y de sus gráficos de mayor resolución. En el 64 la zona de juego se ha hecho considerablemente más pequeña que el tamaño de pantalla disponible, de modo que concuerde exactamente con la versión para el Spectrum.

Pero ambas versiones son, sin duda alguna, divertidas. El ritmo del juego y la dificultad de los problemas planteados se han elaborado con sumo cuidado, haciendo que el juego resulte cautivador para el usuario. Y ahora Matthew Smith ha producido una continuación, *Jet Set Willy*, que rápidamente se está creando su propio culto, tal como ha sucedido antes con *Manic miner*.

Manic Miner: Para el Spectrum de 48 K y el Commodore 64
Editado por: Software Projects, Bear Brand Complex, Allerton Road, Woolton, Liverpool, Merseyside L25 7SF
Autor: Matthew Smith
Palancas de mando: Ambas versiones
Formato: Cassette



"Manic miner" en el Spectrum



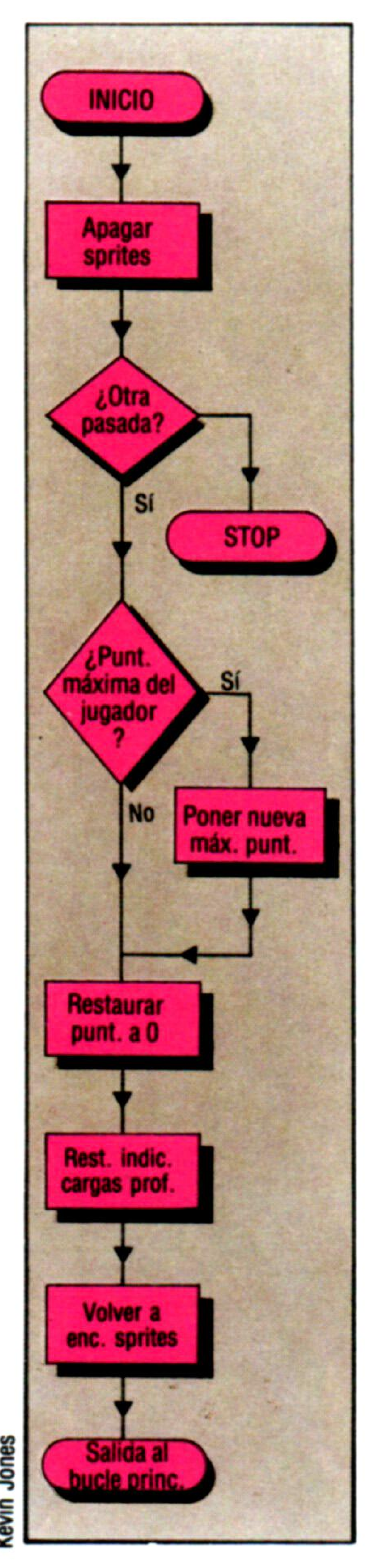
"Manic miner" en el Commodore 64

Bajo tierra
Los misteriosos y maravillosos objetos con que uno se encuentra en el mundo subterráneo de *Manic miner* han contribuido a que el juego sea objeto de multitudinaria aceptación. Los jugadores ya experimentados suelen jactarse de los objetos más insólitos que han llegado a descubrir en las cavernas

Ian McKinnell

Cargas mortíferas

Escribimos hoy las rutinas que crean una explosión cuando el submarino es alcanzado por una carga de profundidad y explicamos el final del juego



Kevin Jones

En el penúltimo capítulo del curso descubrimos lo fácil que es detectar colisiones entre los sprites utilizando un registro de colisión de sprites, V + 30. Cuando esto sucede, la subrutina HIT (que comienza en la línea 5000) tiene que realizar tres tareas. En primer lugar, debe producir una explosión en el punto de la pantalla donde han colisionado los dos sprites, y luego debe incrementar el marcador del jugador según el valor del submarino, que se calcula a partir de su velocidad (DX) y su profundidad (Y3). Por último, debe restaurar las coordenadas para que el próximo submarino comience a desplazarse a través de la pantalla. Analicemos el código de la subrutina HIT (líneas 5000-5250).

La línea 5010 coloca (POKE) un cero en el registro de colisión V + 30 para limpiarlo. Commodore sostiene que el registro de colisión de sprites se limpia él solo después de que dos sprites hayan pasado el uno sobre el otro y ya no estén en colisión. No obstante, la experiencia demuestra que el registro no siempre se limpia él solo con suficiente rapidez, creando efectos inesperados tales como que se produzcan explosiones sin ningún motivo. La solución consiste en limpiar manualmente el registro de colisión después que haya una. Hecho esto, se puede posicionar y encender el sprite de la explosión.

La línea 5030 le da a la explosión una coordenada X de diez pixels a la derecha con respecto a la de la carga de profundidad. Esta ligera desviación sitúa la explosión más centrada respecto a las cargas de profundidad. Como X2 toma su valor a partir de la coordenada X del barco (X0), su valor tiene un límite máximo de 245. Esto significa que el valor máximo de la coordenada X de la explosión es 255. La coordenada Y para la explosión se toma directamente de la del submarino.

Al sprite de la explosión se lo ha designado sprite 1. La línea 5040 pone el bit 1 del registro V + 21 a uno, encendiendo el sprite 1 sin perturbar los valores de otros bits del registro. Llegados a este punto es interesante notar que el sprite de la explosión aparecerá encima de los sprites del submarino y la carga de profundidad, o frente a ellos. Esto se conoce como *prioridad de sprites*, y se rige por la sencilla regla de que los sprites de número inferior aparecen sobre los que llevan un número superior.

El color del sprite de la explosión se controla mediante la posición V + 40 del chip VIC. Se puede obtener un efecto interesante utilizando un bucle FOR...NEXT para colocar (POKE) números de código de color entre 1 y 15. Un bucle FOR...NEXT exterior repite este proceso 20 veces (líneas 5060-5100). Cuando la explosión se ha completado, los tres sprites (explosión, cargas de profundidad y submarino) deben desaparecer de la pantalla. La línea 5130 apaga los sprites 1, 2 y 3.

Como ya hemos mencionado anteriormente, es necesario actualizar la puntuación del jugador utilizando la subrutina que comienza en la línea 5500. Dado que la puntuación se ha de incrementar en función del valor del submarino (en vez de disminuirla, como ocurre cuando un submarino alcanza ileso el borde derecho de la pantalla), el valor de DS se pone a uno para señalarla. Por último, antes de que otro sumergible pueda viajar a través de la pantalla, es necesario restaurar sus coordenadas empleando la subrutina de la línea 2500 y se debe volver a encender el sprite del submarino. Además, el indicador que señala el lanzamiento de una carga de profundidad se debe restaurar a cero de modo que el jugador pueda comenzar a disparar cargas de profundidad nuevamente.

Al cabo de tres minutos el programa abandona el bucle principal y salta a la línea 400. Cuando analizamos por primera vez la utilización del reloj del Commodore 64 (véase p. 714), la línea 400 era una simple sentencia END. La rutina Final del Juego permite volver a jugar el juego y grabar las puntuaciones máximas. El diagrama de flujo muestra las tareas que se han de incorporar en esta rutina. En el listado del programa, estas tareas se llevan a cabo entre las líneas 400 y 660. La mayor parte del código es autoexplicativo, recordando que CHR$(19) lleva el cursor hasta la esquina superior izquierda de la pantalla y CHR$(144) hace que las siguientes letras que se impriman (PRINT) se coloreen en negro.

En este corto proyecto de programación para el Commodore 64 hemos aprendido a crear un sencillo juego animado. Al construir el programa hemos cubierto los aspectos principales que intervienen en la programación de esta clase de juegos en BASIC. Quizá usted desee agregarle al programa refinamientos por su propia cuenta, aplicando los principios que hemos aprendido (p. ej., incorporando los cuatro sprites que han quedado sin utilizar).

Tabla de las variables utilizadas en el Sunhunter	
V	Comienzo de los registros del chip VIC
FL	Indic. cargas prof.; se pone a 1 si se arroja una carga
SC	Puntuación actual del jugador
HS	Máxima puntuación hasta el momento
TI$	Reloj del propio Commodore 64
X0	Coordenada X del barco
X2,Y2	Coordenadas X e Y de la carga de profundidad
X3,Y3	Coordenadas X e Y del submarino
H3,L3	Byte *hi* y byte *lo* de la coordenada X del submarino
DX	Número de pixels; en razón de él se incrementa coord. X
DS	Indicador para ver si se ha de aumentar (DS = 1) o disminuir un marcador



"Subhunter": el listado final

```
10 REM ****************************************************
30 REM **PROYECTO PROGRAMACION 64**
70 REM ****************************************************
90 POKE55,0:POKE56,48:CLR: REM BAJAR TOPE MEM
100 V = 53248:FL = 0:SC = 0
110 GOSUB1000: REM CREACION PANTALLA
120 GOSUB2000: REM CREACION SPRITES
130 GOSUB2500: REM ESTABLECER COORD SUB
140 TI$ = "000000"
200 REM ****BUCLE PRINCIPAL****
210 REM **RELOJ**
220 PRINTCHR$(19);:PRINTTAB(14)CHR$(5);"TIEMPO"MID$(TI$,3,2)"
    RIGHT$(TI$,2)
225 IFVAL(TI$) > 259 THEN 400: REM FINAL JUEGO
230 GET A$
240 IF A$ = "Z" THEN X0 = X0-1.5:IF X0 < 24 THENX0 = 24
250 IF A$ = "X" THEN X0 = X0 + 1.5:IF X0 > 245 THENX0 = 245
260 IF A$ = "M" AND FL = 0 THEN GOSUB3000: REM PREPARAR
    CARGAS PROFUNDIDAD
270 REM **MOVER BARCO**
290 POKE V,X0
300 REM **MOVER SUB**
310 X3 = X3 + DX
320 REM **SI EL SUB LLEGA AL BORDE DE LA PANTALLA**
330 IF X3 > 360 THEN DS = -1:GOSUB5500:GOSUB2500
340 H3 = INT(X3/256):L3 = X3-256*H3
350 POKE V + 6,L3
360 IF H3 = 1 THEN POKE V + 16,PEEK(V + 16)OR8:GOTO380
370 POKE V + 16,PEEK (V + 16)AND247
380 IF FL = 1 THEN GOSUB4000:REM MOVER CARGA PROFUNDIDAD
390 GOTO 200: REM RECOMENZAR BUCLE PRINCIPAL
400 REM ****FIN DE CONDICIONES DEL JUEGO****
410 REM **APAGAR SPRITES**
420 POKE V + 21,0
430 REM **RESTAURAR COORD SUB & BARCO**
440 X0 = 160:GOSUB 2500
450 INPUT"OTRA PARTIDA?(S/N)";AN$
460 IF AN$ <> "S" THENEND
480 REM **BORRAR MENSAJE**
490 PRINT CHR$(19):REM PONER CURSOR EN SU SITIO
500 FOR I = 1 TO 120
510 PRINT"   ";
520 NEXT I
540 REM **ESTABLECER MAX PUNTUACION**
550 IF SC > HS THEN HS = SC
560 PRINT CHR$(19);CHR$(144);"PUNTUACION 000":SC = 0
570 PRINT CHR$(19);
580 PRINT TAB(26);CHR$(144);" MAX PUNTUACION;HS
600 REM **RESTAURAR RELOJ E INDICADOR**
610 TI$ = "000000":FL = 0
630 REM **ENCENDER SUB & BARCO**
640 POKE V + 21,9
660 GOTO200: REM RECOMENZAR BUCLE
1000 REM ****CREACION PANTALLA****
1010 PRINT CHR$(147): REM LIMPIAR PANTALLA
1030 REM **COLOR MAR**
1040 POKE 53281,14:POKE 53280,6
1050 FOR I = 1264 TO 1943
1060 POKE I,160:POKE I + 54272,6
1070 NEXT
1090 REM **FONDO MAR**
1100 FORI = 1944 TO 2023
1110 POKE I,102:POKE I + 54272,9
1120 NEXT
1130 POKE 650,128:REM REPETIR TECLAS
1150 REM **PUNTUACION**
1160 PRINT CHR$(19);CHR$(144);"PUNTUACION 000";SPC(16);"MAX
     PUNTUACION 000"
1170 RETURN
2000 REM ****CREACION SPRITES****
2020 REM **LEER DATOS BARCO**
2030 FOR I = 12288 TO 12350
2040 READ A:POKE I,A:NEXT I
2060 REM **LEER DATOS SUB**
2070 FOR I = 12352 TO 12414
2080 READ A:POKE I,A:NEXT
2100 REM **LEER DATOS CARGAS**
2110 FOR I = 12416 TO 12478
2120 READ A:POKE I,A:NEXT
2140 REM **LEER DATOS EXPLOSION**
2150 FOR I = 12480 TO 12542
2160 READ A:POKE I,A:NEXT
2180 REM **ESTABLECER PUNTEROS**
2190 POKE 2040,192:POKE 2041,193: POKE 2042,194
2200 POKE2043,195
2220 REM **ESTABLECER COLORES**
2230 POKE V + 39,0:POKE V + 40,1:POKE V + 41,0
2240 POKE V + 42,0
2260 REM **ESTABLECER COORD INICIALES**
2270 POKE V + 1,80:X0 = 160: REM COORD BARCO
2280 POKE V + 29,15:POKE V + 23,2
2300 REM **ENCENDER SPRITES 0 & 3**
2310 POKE V + 21,9
2320 RETURN
2500 REM ****RESTAURAR COORD SUB****
2510 Y3 = 110 + INT(RND(TI)*105)
2520 POKE V + 7,Y3:POKE V + 6,0
2530 X3 = 0:DX = RND(TI)*3 + 1
2540 POKE V + 16,0
2550 RETURN
3000 REM ***PREPARAR CARGAS PROFUNDIDAD****
3020 REM **PONER INDICADOR**
3030 FL = 1
3050 REM **ESTABLECER COORD**
3060 Y2 = 95:X2 = X0
3070 POKE V + 4,X2:POKE V + 5,Y2
3090 REM **ENCENDER SPRITE 2**
3100 POKE V + 21,PEEK(V + 21)OR4
3110 RETURN
4000 REM ****MOVER CARGA PROFUNDIDAD****
4020 REM **DECREMENTAR COORD Y**
4030 Y2 = Y2 + 2
4050 REM **VERIFICAR FONDO MAR & APAGAR**
4060 IF Y2 > Y3 + 25 OR Y2 > 216 THEN
     POKEV + 21,PEEK(V + 21)AND251:FL = 0
4070 POKE V + 5,Y2
4090 REM **VERIFICAR SI SUB ACERTADO**
4100 IF PEEK(V + 30) = 12 THEN GOSUB 5000:REM
     RUTINA HIT
4110 RETURN
5000 REM ****RUTINA HIT****
5010 POKE V + 30,0:REM LIMPIAR REG COLISIONES
5020 REM **ENCENDER SPRITE EXPLOSION**
5030 POKE V + 2,X2 + 10:POKE V + 3,Y3
5040 POKE V + 21,PEEK(V + 21)OR2
5060 REM **DESTELLOS COLORES**
5070 FOR I = 1 TO 20
5080 FOR J = 1 TO 15
5090 POKE V + 40,J
5100 NEXT J:NEXT I
5120 REM **APAGAR SPRITES 1, 2 & 3**
5130 POKE V + 21,PEEK(V + 21)AND241
5150 REM **ACTUALIZAR PUNTUACION**
5160 DS = 1:GOSUB 5500
5180 REM **RESTAURAR COORD SUB & INDIC.**
5190 FL = 0:GOSUB 2500
5210 REM **VOLVER A ENCENDER SUB**
5220 POKE V + 21,PEEK(V + 21)OR8
5230 RETURN
5500 REM ****ACTUALIZAR PUNTUACION****
5510 SC = SC + INT(Y3 + DX*30)*DS
5520 IF SC < 0 THEN SC = 0
5530 PRINT CHR$(19);CHR$(144)"PUNTUACION";
     SC;CHR$(157);"  "
5540 RETURN
6000 REM ****DATOS BARCO****
6010 DATA0,0,0,0,0,0,0,0,0
6020 DATA0,128,0,0,192,0,0,192,0
6030 DATA0,192,0,1,224,0,1,224,0
6040 DATA13,224,0,3,248,128,3,253,8
6050 DATA15,254,16,31,255,48,255,255,255
6060 DATA127,255,254,63,255,254,31,255,252
6070 DATA0,0,0,0,0,0,0,0,0
6100 REM ****DATOS EXPLOSION****
6110 DATA0,0,0,0,0,0,0,16,0,0,8,0,4,16
6120 DATA0,3,2,64,1,56,128,12,255,144
6130 DATA1,238,40,5,151,0,11,121,0,1
6140 DATA183,0,25,214,96,0,236,48,6,24
6150 DATA152,3,98,0,8,51,0,0,96,128,0
6160 DATA64,0,0,0,0,0,0,0
6170 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0
6200 REM ****DATOS CARGAS PROFUNDIDAD****
6210 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
6220 DATA0,0,0,32,0,0,32,0,0,32,0,0,32,0
6230 DATA0,0,0,0,0,0,0
6240 DATA2,0,0,2,0,0,2,0,0,2,0,0
6250 DATA0,0,0,0,0,0,0,0
6260 DATA0,0,0,0,0,0,0,0
6300 REM ****DATOS SUBMARINO****
6310 DATA0,0,0,0,0,0,0,0,0,0,0,0
6320 DATA0,8,0,0,12,0,0,12,0
6330 DATA0,12,0,0,28,0,0,60,0
6340 DATA0,126,0,199,255,255
6350 DATA239,255,255,127,255,255
6360 DATA255,255,254,199,255,254
6370 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Éste es el listado final para nuestro programa Subhunter junto con una tabla de las variables empleadas en el mismo. El listado contiene muchas sentencias REM para facilitar su comprensión. Éstas se pueden pasar por alto al entrar el programa en el ordenador propio, pero hay que tener cuidado en no eliminar alguna línea REM que se requiera en otra parte del programa. Por ejemplo, se podría optar por eliminar el REM de la línea 400, pero este número de línea se utiliza como parte de una sentencia GOTO en la línea 225. Si elimina la línea 400 por completo aparecería un mensaje "ERROR DE SENTENCIA NO DEFINIDA EN LA LINEA 225" (UNDEF'D STATEMENT ERROR AT LINE 225) y el programa se colgaría. La mejor forma de evitar esto consiste en omitir sólo aquellas REM que aparezcan al final de una línea y aquellas líneas que utilicen dos puntos (:) para separar las instrucciones

Dividir y ver

Con este capítulo damos fin al estudio básico del lenguaje máquina. En él explicaremos cómo se divide y cómo se programa la visualización por pantalla

Lo mismo que nos ocurrió con la multiplicación al estilo tradicional, que nos sirvió de modelo algorítmico para la multiplicación en sistema binario (véase p. 778), nos pasa ahora con la división. Consideremos el siguiente ejemplo:

```
1 0 0 1 1 0 1 0  |1011              dividendo-divisor
-1 0 1 1 ↓        00001110                 cociente
0 1 0 0 0 0
  -1 0 1 1 ↓                       restas sucesivas
0 0 0 1 0 1 1                             del divisor
     -1 0 1 1 ↓
0 0 0 0 0 0 0 0                               resto
```

El meollo de este algoritmo está en cómo resta una y otra vez el divisor a los bits superiores que van quedando del dividendo. Según sea el resultado, se va colocando en el cociente un 1 o un 0. El resto es el resultado de la última sustracción.

Por desgracia, no son tan fáciles de explicar las diversas maneras de adaptar este método en lenguaje máquina, como lo fueron en la multiplicación. Digamos al menos que el Z80 no deja de aprovechar el potente y flexible recurso que le brindan sus registros de 16 bits, mientras que el 6502 sigue con su tratamiento de 8 bits por vez.

El divisor se sitúa en la dirección etiquetada con DIVSR, el dividendo en DVDND, el cociente (inglés, *quotient*) en QUOT, y el resto *(remainder)* en RMNDR. A continuación pasaremos, sin más, a ofrecerle el programa en assembly, tanto para el Z80 como para el 6502.

Observe cómo en ambos casos el divisor resta al dividendo parcial y, si el resultado es negativo, cómo se recupera este dividendo sumándole de nuevo el divisor. En el 6502 es notable el empleo que se hace del registro de estado del procesador (PSR) tras la resta del divisor: hay un desplazamiento rotatorio del flag de arrastre hacia el cociente, sin llegar a perder su estado, pues se necesita para indicar además el resultado de la resta. Por eso es depositado el PSR en la pila antes de la rotación, para ser recuperado posteriormente, devolviendo al flag el estado que tomó después de la resta.

Concluimos así el análisis de las cuatro reglas aritméticas. Usted habrá comprobado su interés pedagógico como medio de estudio, pues nos revela cómo trabaja la máquina, pero no vale la pena entretenerse mucho más en ellas. Entre otras razones, porque los programadores hace tiempo que han puesto a nuestra disposición, tanto en revistas como en libros de texto las diversas rutinas que se suelen usar. Si, más adelante, necesitáramos alguna de ellas, se la proporcionaremos o la expondremos en algún ejercicio.

DIVISIÓN DE 16 BITS POR 8 BITS

Z80			6502		
START	LD	A,(DIVSR)	START	LDA	#$00
	LD	D,A		STA	QUOT
	LD	E,$00		STA	RMNDR
	LD	HL,(DVDND)		LDX	#$08
	LD	B,$08		LDA	DVDHI
LOOP0	AND	A		SEC	
	SBC	HL,DE		SBC	DIVSR
	INC	HL	LOOP0	PHP	
	JP	P,POSRES		ROL	QUOT
NEGRES	ADD	HL,DE		ASL	DVDLO
	DEC	HL		ROL	A
POSRES	ADD	HL,HL		PLP	
	DJNZ	LOOP0		BCC	CONT1
	LD	(QUOT),HL		SBC	DIVSR
	RET			JMP	CONT2
DIVSR	DB	$F9	CONT1	ADC	DIVSR
DVDND	DW	$FDE8	CONT2	DEX	
QUOT	DB	$00		BNE	LOOP0
RMNDR	DB	$00		BCS	EXIT
				ADC	DIVSR
				CLC	
			EXIT	ROL	QUOT
				STA	RMNDR
				RTS	
			DIVSR	DB	$F9
			DVDLO	DB	$E8
			DVDHI	DB	$FD
			QUOT	DB	$00
			RMNDR	DB	$00

La pantalla

Hasta aquí nos hemos limitado a tratar la memoria RAM y la CPU como un sistema de cálculo y los resultados los íbamos dejando en la misma RAM, en algún lugar, disponibles para ser visualizados por medio de un programa monitor. Esto no basta, desde luego, pero tampoco había necesidad alguna de ocuparnos de la pantalla cuando lo que nos interesaban eran las subrutinas aritméticas.

Muchos micros cuentan con una imagen en memoria de la pantalla. Esto quiere decir que hay reservada un área de la RAM para retener la imagen de la pantalla. Cualquier visualización se realiza a base de puntos o *pixels*, que están o encendidos o apagados. Son, pues, representables por medio de unos (encendido) y ceros (apagado), en binario, y el contenido total de la pantalla puede considerarse como un "mapa" de puntos que se corresponden con los bits que componen los bytes pertenecientes a la RAM de pantalla. Lástima que esta técnica

empleada tanto por el BBC Micro como por el Spectrum y el Commodore 64 no sea utilizada por ninguno de ellos de manera inmediata. Lo más fácil para nosotros sería que cada fila de la pantalla quedara determinada por bytes correspondientes a pixels numerados consecutivamente de izquierda a derecha, siendo el byte más extremo por la izquierda el que sigue al byte extremo derecho de la fila anterior. Pero esto no ocurre así, por varias razones. Lo que significa que debemos examinar cada máquina por separado.

El Spectrum funciona siempre en alta resolución y dispone de un área de la memoria para la imagen de la pantalla. Esta correspondencia memoria-para-pantalla con pantalla es compleja, pues la pantalla se divide en tres bloques horizontales de ocho filas de PRINT cada uno, y a su vez cada una de estas filas se divide en otras ocho filas de pixels. Dentro de ellas el direccionamiento de los bytes que las componen es secuencial, pero no así de una fila a otra. El BBC Micro y el Commodore 64 no siguen este esquema, pero resultan tanto o más intrincados. Pensamos que la mejor manera de entenderlos es estudiar de momento sólo la forma en que hacen aparecer en pantalla los caracteres ASCII. Veámoslo.

Al tratarse de una tarea de uso frecuente, sus correspondientes rutinas en lenguaje máquina se hallan en la ROM. Contando con una descripción detallada de cómo operan, tales rutinas pueden ser llamadas desde nuestros programas en assembly. Necesitamos conocer tan solo la dirección de llamada, los registros de comunicación y cualquier otro requisito previo.

El Spectrum prescinde de todo paso preliminar: el acumulador hace de registro comunicador, y allí debemos colocar el código ASCII del carácter y visualizar. Basta después con usar la instrucción RST $10 para obtener en pantalla en el lugar indicado por el cursor, el carácter correspondiente a dicho código. Algo así es lo que exigen los otros dos sistemas, sólo que el opcode RST (ReSTart: volver al inicio) es una instrucción típica del Z80; se trata de una instrucción bifurcadora relacionada con la página cero y, por ello, implementada con un solo byte: admite uno de los ocho operandos siguientes: $00, $08, $10, $18, $20, $28, $30 y $38. En algún sitio de la página cero ha de hallarse la dirección de inicio de una rutina ROM, y a esta dirección es hacia la que apuntarán los posibles operandos. Tales rutinas toman (o llevan) caracteres o (o de) la pantalla, y no son llamadas directamente por su dirección de comienzo sino a través de la instrucción RST. Esto se debe, en parte, a razones de rapidez (pues, aunque sólo la CPU note la diferencia, es más rápido usar RST que CALL); y en parte, previendo la portabilidad del programa. Si todo programador del Z80 sabe que RST $10, por ejemplo, está llamando a la rutina PRINT, ninguno de ellos se molestará en saber dónde situó el ingeniero de una máquina que incorpore el Z80 esta rutina, por lo que dicho ingeniero podrá colocarla donde más le interese, con tal de que la página cero quede dispuesta de manera que las posiciones de RST conduzcan el programa hasta las direcciones de las rutinas convenidas. El procedimiento es muy semejante para el BBC Micro. El carácter cuyo código se ha colocado en el acumulador aparecerá en pantalla, en la posición que indica el cursor, mediante la instrucción JSR $FFEE. Estamos ante la conocida rutina OSWRCH, perfectamente descrita en la *Guía avanzada del usuario*.

El Commodore 64 también sigue este esquema. Para visualizar el carácter cuyo código ASCII está en el acumulador se emplea JSR $FFD2. Es la rutina CHKOUT, y su descripción se encuentra en la *Guía de referencia del programador*.

Éste es, pues, el esquema general para el empleo de las rutinas de la ROM, que a la vez nos ilustra el funcionamiento de los registros de comunicación. Tiene doble sentido toda comunicación entre el programa que llama a una rutina y la rutina llamada. Por ejemplo, una rutina que sirve para tomar información externa (rutina input) toma dicha información desde el dispositivo externo y lo lleva a la CPU siempre a través del acumulador. Y si falla el proceso, la misma subrutina se sirve de uno de los registros para devolver un código de error. Todos estos protocolos se explican exhaustivamente en muchas obras de referencia hoy en el mercado, dedicadas a exponer las características específicas de cada máquina.

En capítulos venideros trataremos este tema de la recogida de información desde el teclado u otro periférico y estudiaremos igualmente el trazado en alta resolución desde lenguaje máquina. Resumiremos a continuación el camino que ya hemos recorrido hasta este momento en nuestro curso de lenguaje máquina.

En resumen

Comenzamos el curso dando una idea general del lenguaje máquina y tratando de quitarle la mística que habitualmente le envuelve. Por eso insistimos en considerarlo un lenguaje más entre los utilizados por nosotros y por las máquinas. Vimos cómo una misma secuencia de bytes dentro de la RAM podía representar un string de caracteres ASCII en un momento dado y a renglón seguido ser interpretados como una sentencia de BASIC, o bien una serie de direcciones de dos bytes, o bien una secuencia de instrucciones en lenguaje máquina. Por poco que se entretuviera practicando con un programa monitor de lenguaje máquina, éste ya le habrá evidenciado que una secuencia de bytes pueden ser "desensamblados" como tres secuencias de instrucciones tan diferentes como válidas, a poco que a usted le de por comenzar a partir del primero, segundo o tercer byte de la secuencia. No hay nada en este lenguaje que impida tal hecho, y, además, la misma CPU no es capaz de indicarle si lo que está ejecutando es lo que usted escribió o bien una versión embarullada que se metió de rondón en la memoria.

Consideramos después la distribución de la memoria y los modos convencionales de direccionamiento. Para entender todo esto no tuvimos más remedio que estudiar la aritmética en sistema binario, lo que nos permitió conocer más a fondo la CPU: vimos que los procesadores de sólo ocho bits, salvo algunas excepciones, no nos permitían más que el tratamiento de un byte por vez (o sea, de los números decimales comprendidos entre el 0 y el 255). Pronto nos dimos cuenta de las limitaciones del sistema decimal dentro del lenguaje assembly, en comparación con el más apropiado sistema binario. Así, tratando el tema de la memoria distribuida

por páginas (paginación) vimos que la numeración de éstas y su tamaño debían estar en función del número-base, el dos, y sus potencias. Dos elevado a ocho es igual a 256, o, por decirlo en otras palabras, el número "mágico" en el sistema de un procesador de ocho bits.

Otro descubrimiento inmediato fue lo difícil que resultaba operar continuamente en binario, cosa que arreglamos pasándonos a los números hexadecimales (numeración en base 16). Observamos como un byte, que son ocho bits, podía escribirse con sólo dos dígitos "hexas" (como familiarmente les llamamos), desde el $00 al $FF, uno de los cuales se corresponde con los cuatro primeros bits, y el otro, por su parte, lo hace con los cuatro restantes bits de un byte.

Después examinamos con todo detalle cómo son almacenados los programas en BASIC dentro del área para programas de la memoria. Una buena parte del sistema operativo (SO) se pudo entender gracias al artificio de encerrar en un solo byte toda una orden que literalmente requiere varios (en inglés este byte es conocido por *token*). Tratamos incluso las marcas de fin de línea, viendo cómo el intérprete se las ingeniaba para decidir dónde acaba y dónde empieza un fragmento de lenguaje. Por su parte, el direccionamiento de enlace típico del Commodore nos presentó otro convencionalismo más: el direccionamiento byte inferior-byte superior (que llamamos *lo-hi*, por el inglés *low-high*) y el direccionamiento indirecto.

A partir de aquí ya estuvimos a punto para encarar el lenguaje assembly. Comenzamos por las operaciones más primitivas de la CPU, originadas por opcodes de ocho bits que son las instrucciones del programa. De esta primera exposición fue fácil pasar al concepto de la mnemotécnica, base del assembly. Y así nos resultó claro que escribir código máquina, o assembly o BASIC era programar y lo que cuenta es saber resolver con lógica un problema antes de darle forma codificada a la solución. Éste fue el tema central del curso. Aunque la oscuridad de algunos conceptos del assembly nos obligó a prestar más atención en aclarar las numerosas confusiones que en la mayoría de los casos generan los lenguajes de bajo nivel en quienes abordan su estudio por vez primera.

Por eso nos extendimos practicando con programas de carga, traslado y manipulación de información útiles dentro de muchos programas en BASIC. Estudiamos también las variables del sistema y los punteros de que disponen el BBC Micro, el Spectrum y el Commodore 64, aprendiendo a "robar" espacio del mismo BASIC.

Examinamos la arquitectura de pequeños sistemas computerizados, las CPU del Z80 y del 6502, pasando ya a escribir programas en assembly que manejaban la memoria y el acumulador. Aquí es donde hablamos de directrices o pseudo-opcodes del ensamblador, que nos acercaban más a la realidad práctica, aun a costa de alejarnos del lenguaje máquina, el ensamblaje manual y la fatigosa programación a bajo nivel.

Necesitábamos en ese momento comprender la construcción lógica de un lenguaje de programación, por lo que presentamos el registro de estado del procesador (el PSR: inglés *Processor Status Register*). La instrucción ADC (sumar con arrastre) que empleamos nos sirvió para explicar cómo este dispositivo servía de registro de los resultados de las operaciones de la CPU. Vimos que el indicador o flag de arrastre era en las operaciones aritméticas una pieza indispensable. Desde este momento no dejamos de hablar del PSR y de las instrucciones que se referían a él.

Echamos una breve ojeada a los diversos modos de direccionamiento, subrayando el direccionamiento indexado por su importancia a la hora de tratar bucles, listas y tablas. Comenzamos entonces a echar de menos las instrucciones capaces de cambiar el flujo de control del programa y así, mientras seguíamos viendo las posibilidades del direccionamiento indexado y del indirecto, iniciamos el estudio de las instrucciones de bifurcación condicionada. Con éstas y con las cuatro reglas aritméticas junto con las estructuras matriciales, observamos que teníamos ya vertebrado cualquier lenguaje de programación. Sólo nos faltaba ahora darle forma por medio de una investigación y de unas prácticas sistemáticas.

El último aspecto del sistema operativo que debíamos estudiar era la pila, y lo hicimos examinando a la vez la manera cómo se llama en assembly a una rutina y cómo se efectúa el retorno (inglés: *call* y *return*). Vimos para qué servía la pila, cómo funciona y cómo usarla, al tiempo que ampliábamos nuestro acopio de trucos para programar en lenguaje máquina y obteníamos una visión más rica de los registros de la CPU y su capacidad para manejar la memoria y el microprocesador.

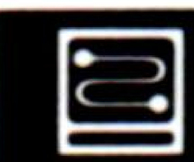
Las postreras lecciones, ya provistos de un buen conocimiento de la arquitectura del microprocesador y de un amplio vocabulario de instrucciones opcodes, fueron dedicadas al estudio de las operaciones aritméticas en binario. No nos arredramos ni ante un concepto tan sutil como el complemento a dos, ni tampoco ante operaciones que resultan tan difíciles de explicar como la resta, la multiplicación o la división.

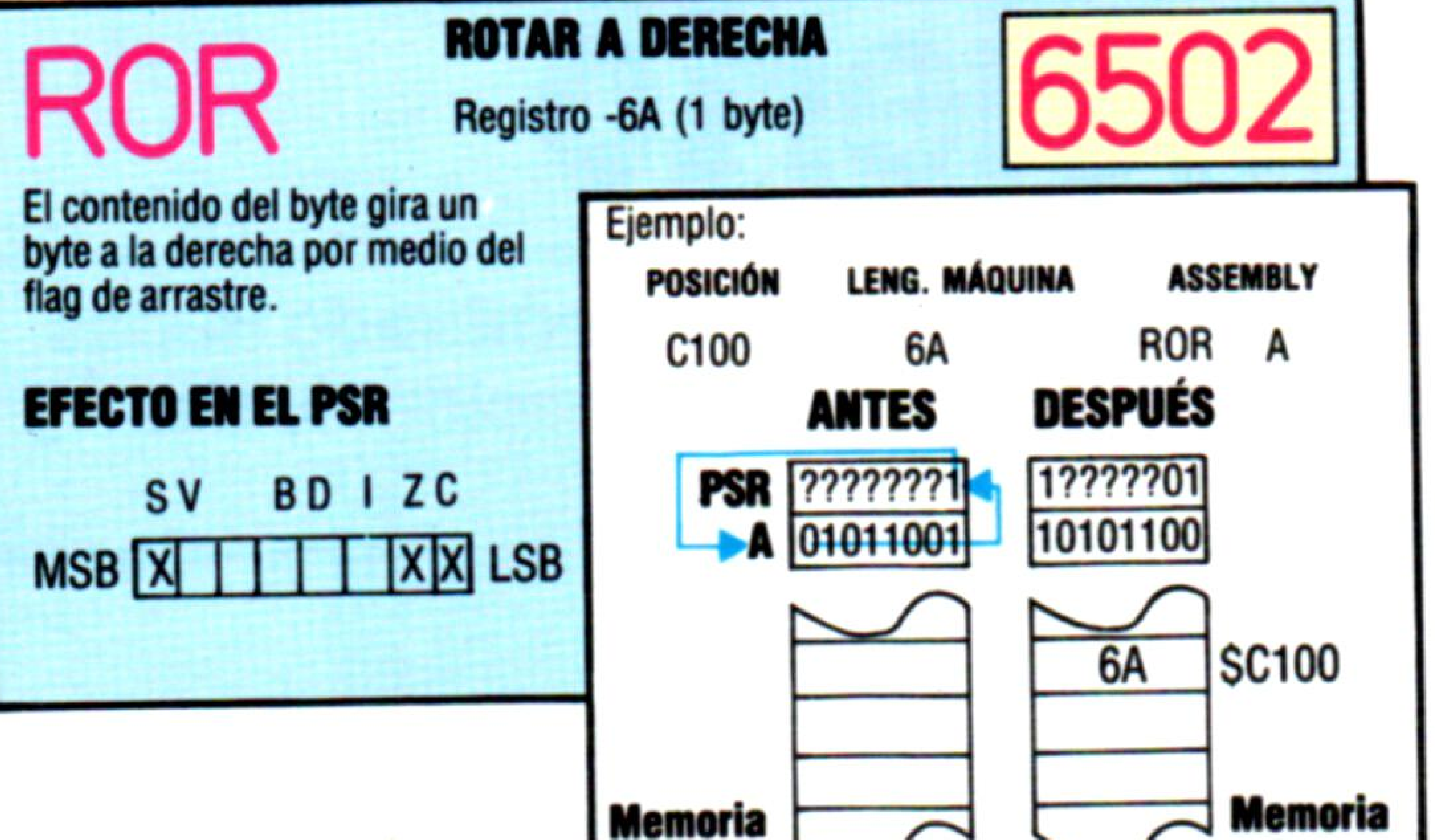
¿Qué nos queda por ver todavía? Ilustrar prácticamente las posibilidades de la programación en lenguaje máquina, a través de tareas específicas que confiaremos tanto a los procesadores que hasta aquí han representado nuestros puntos de apoyo (es decir, el Z80 y el 6502) como a algunos otros, como pueden ser el 6809 CPU, por ejemplo, que utilizan el Dragon 32 y 64.

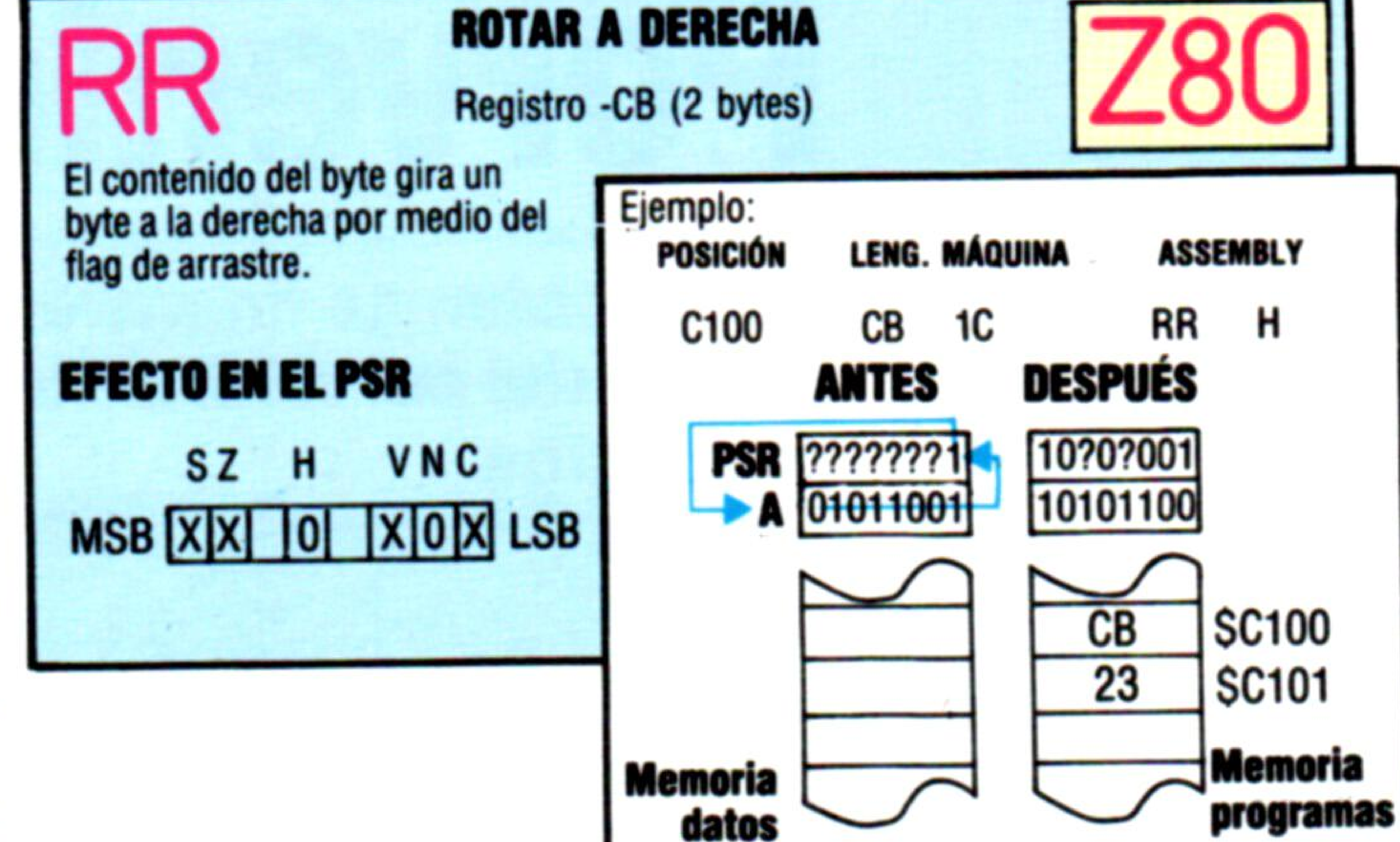
Soluciones a los ejercicios de p. 779

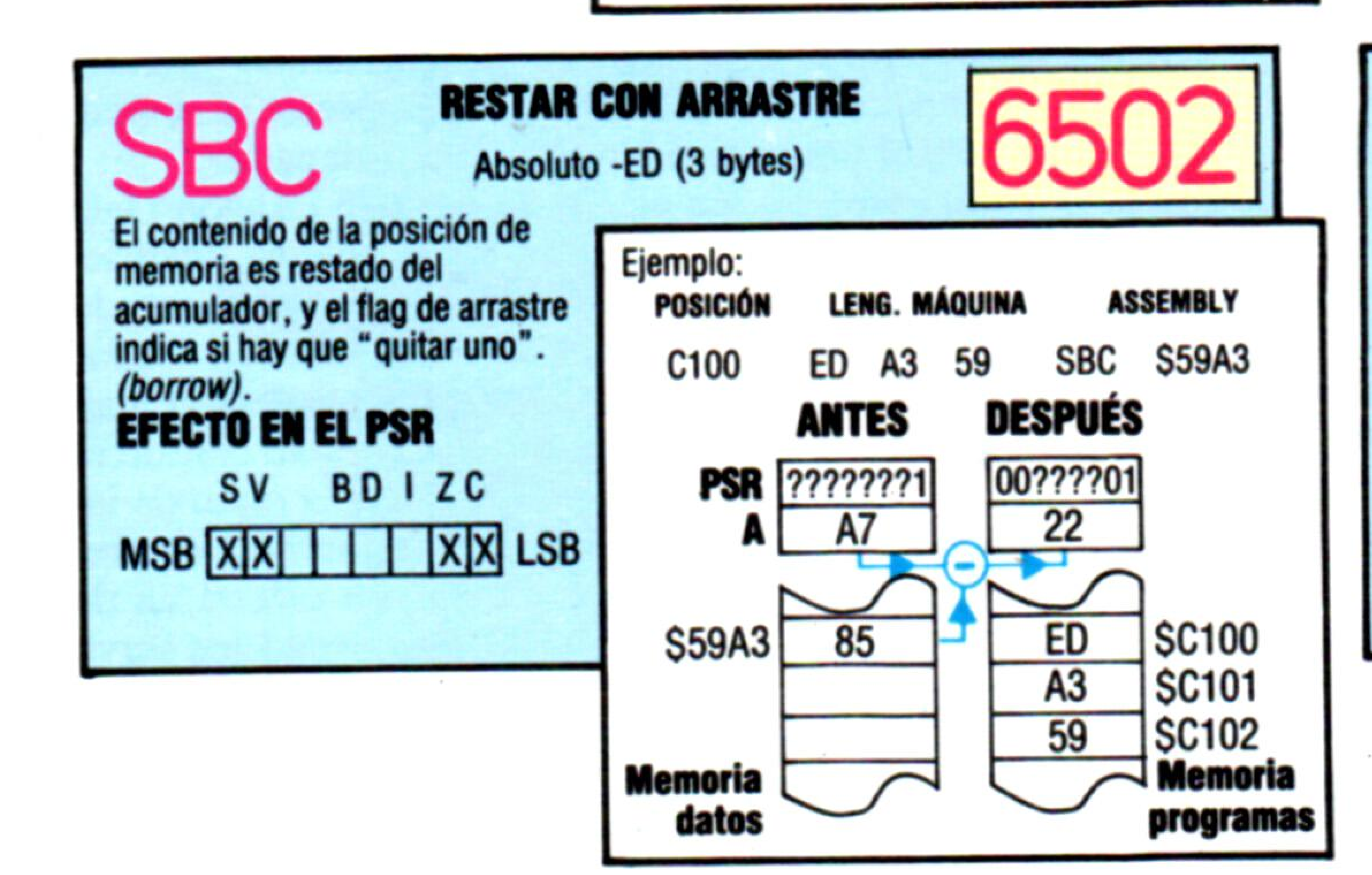
1) La solución de más rápida ejecución es sin duda una rutina que se escriba específicamente para multiplicandos de 16 bits, siguiendo la pauta de la rutina para 8 bits del capítulo anterior. Por otra parte, si usted separa la multiplicación con 16 bits en dos multiplicaciones de 8 bits cada una (al multiplicador del byte *lo* deberá seguir el del byte *hi*), entonces le bastará con hacer una llamada *(call)* por dos veces a esa rutina para 8 bits de que dispone, previendo la posibilidad de un arrastre procedente del byte *lo* y almacenando después el resultado en los bytes reservados al producto.

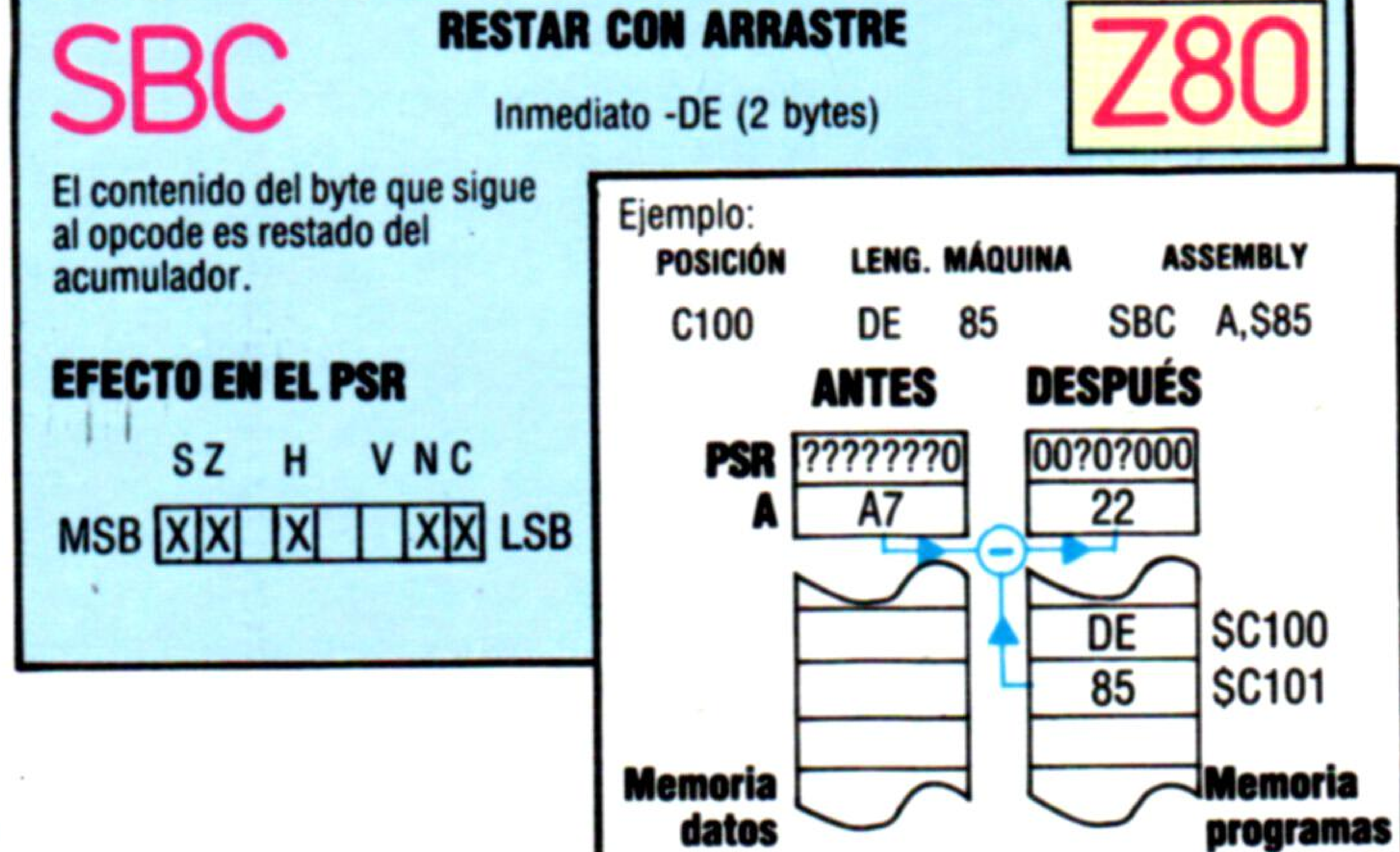
2) Una rutina multiplicadora basada en la adición reiterada consta sencillamente de un bucle cuyo contador vale lo que el multiplicador; cada vez que se ejecuta dicho bucle, el multiplicando se va agregando al producto.

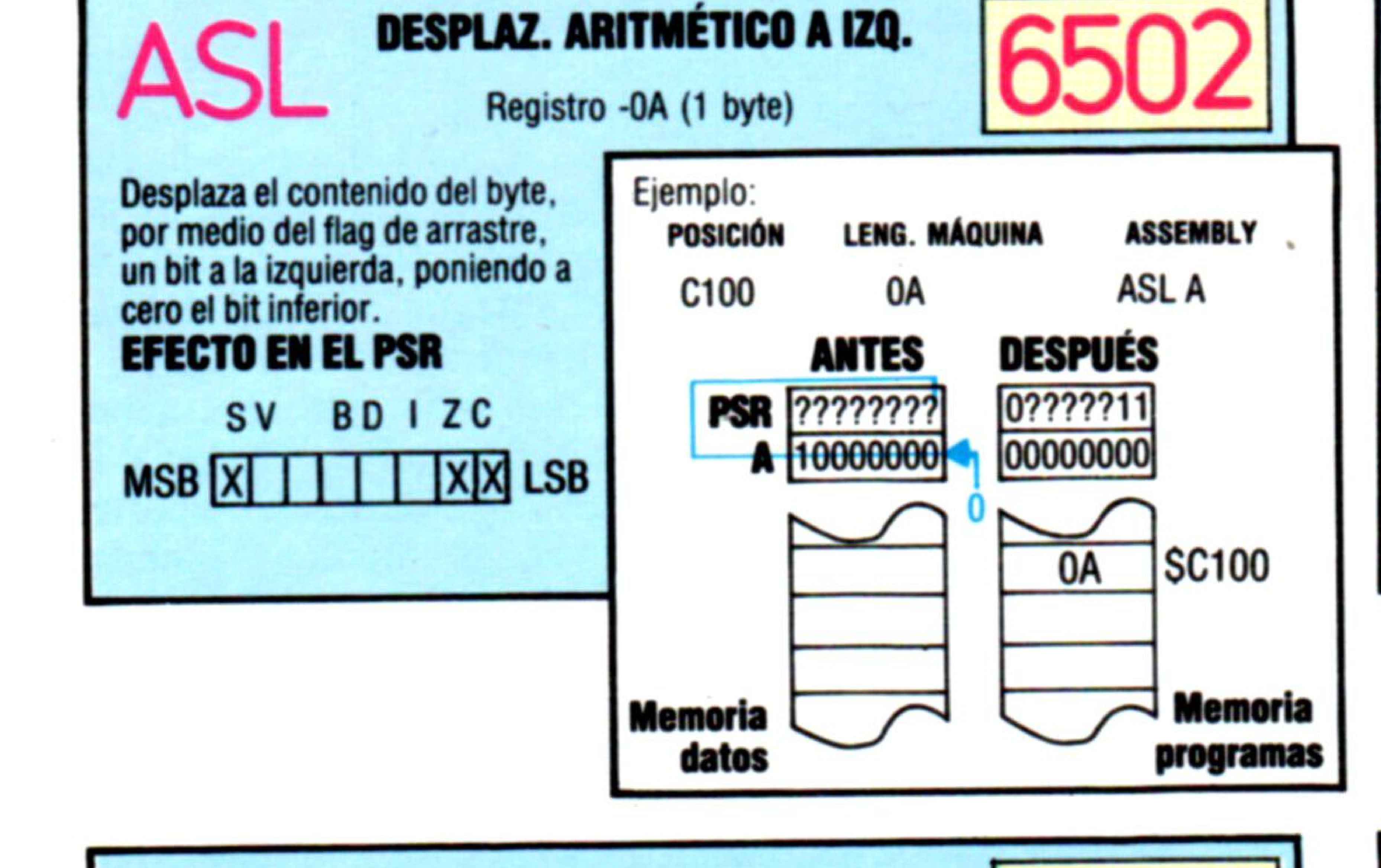


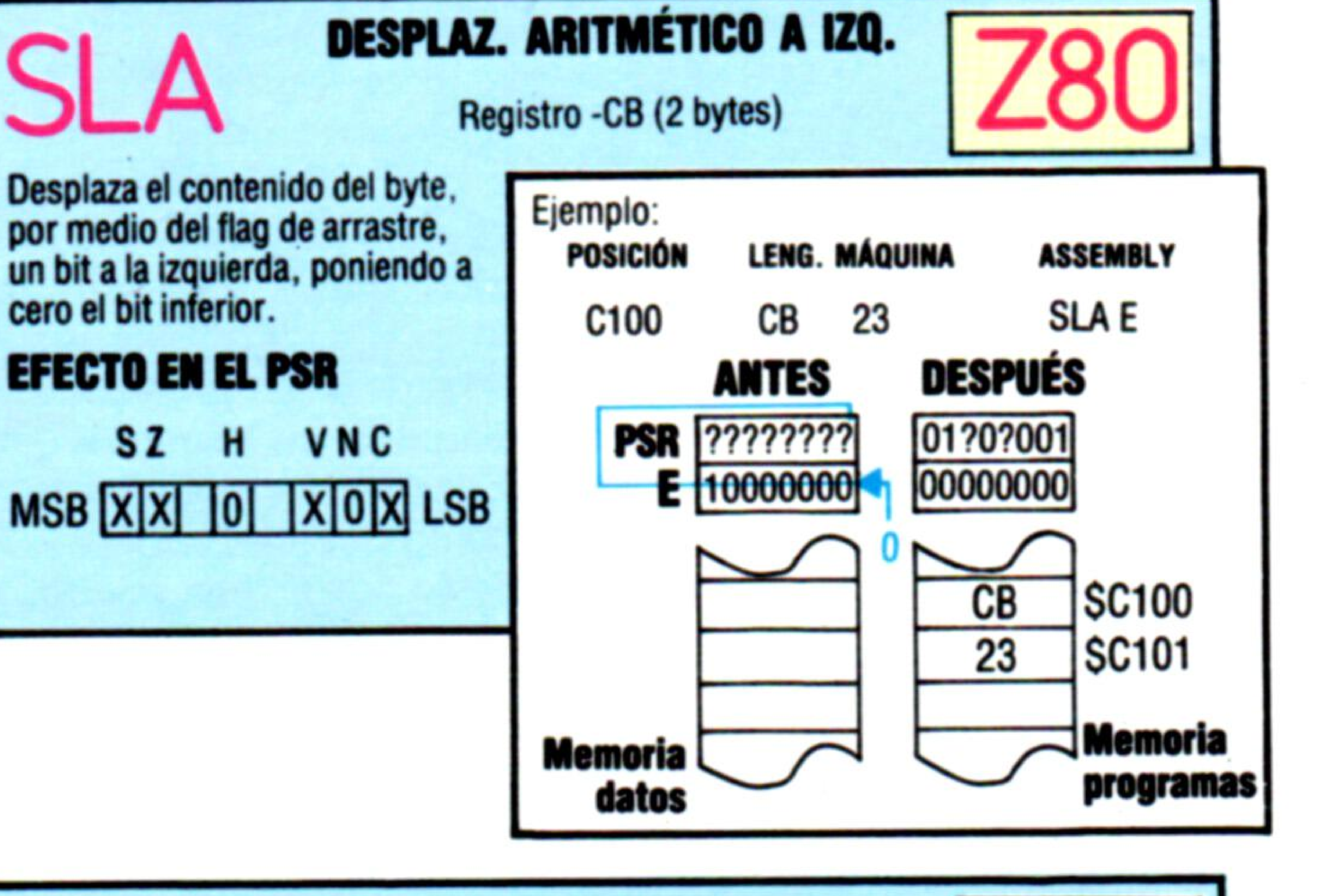

ROR
ROTAR A DERECHA
Registro -6A (1 byte)
6502
El contenido del byte gira un byte a la derecha por medio del flag de arrastre.
EFECTO EN EL PSR
S V BD I Z C
MSB X X X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 6A ROR A
ANTES DESPUÉS
PSR ???????1 1?????01
A 01011001 10101100
6A $C100
Memoria datos
Memoria programas

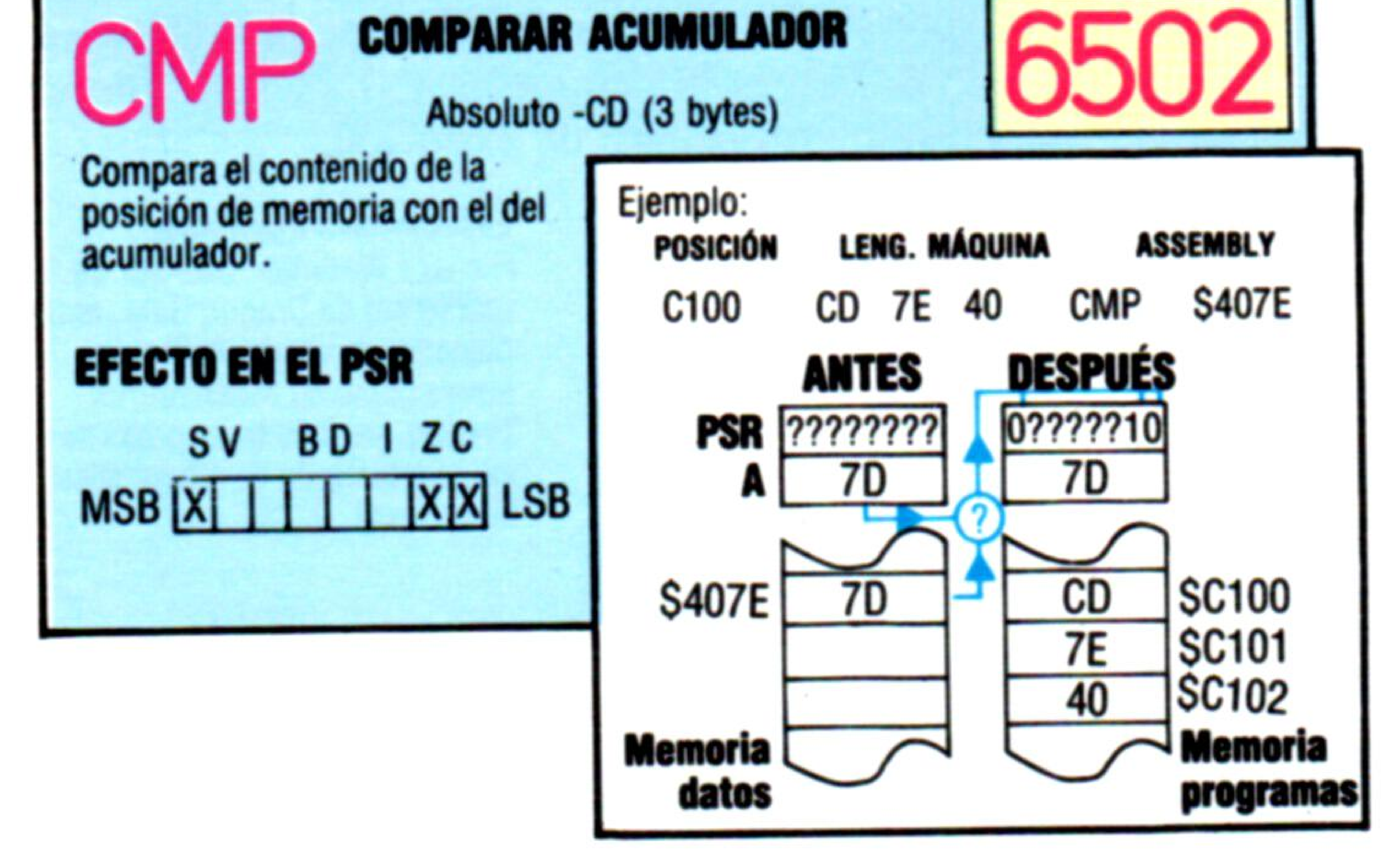

RR
ROTAR A DERECHA
Registro -CB (2 bytes)
Z80
El contenido del byte gira un byte a la derecha por medio del flag de arrastre.
EFECTO EN EL PSR
S Z H V N C
MSB X X 0 X 0 X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 CB 1C RR H
ANTES DESPUÉS
PSR ???????1 10?0?001
A 01011001 10101100
CB $C100
23 $C101
Memoria datos
Memoria programas

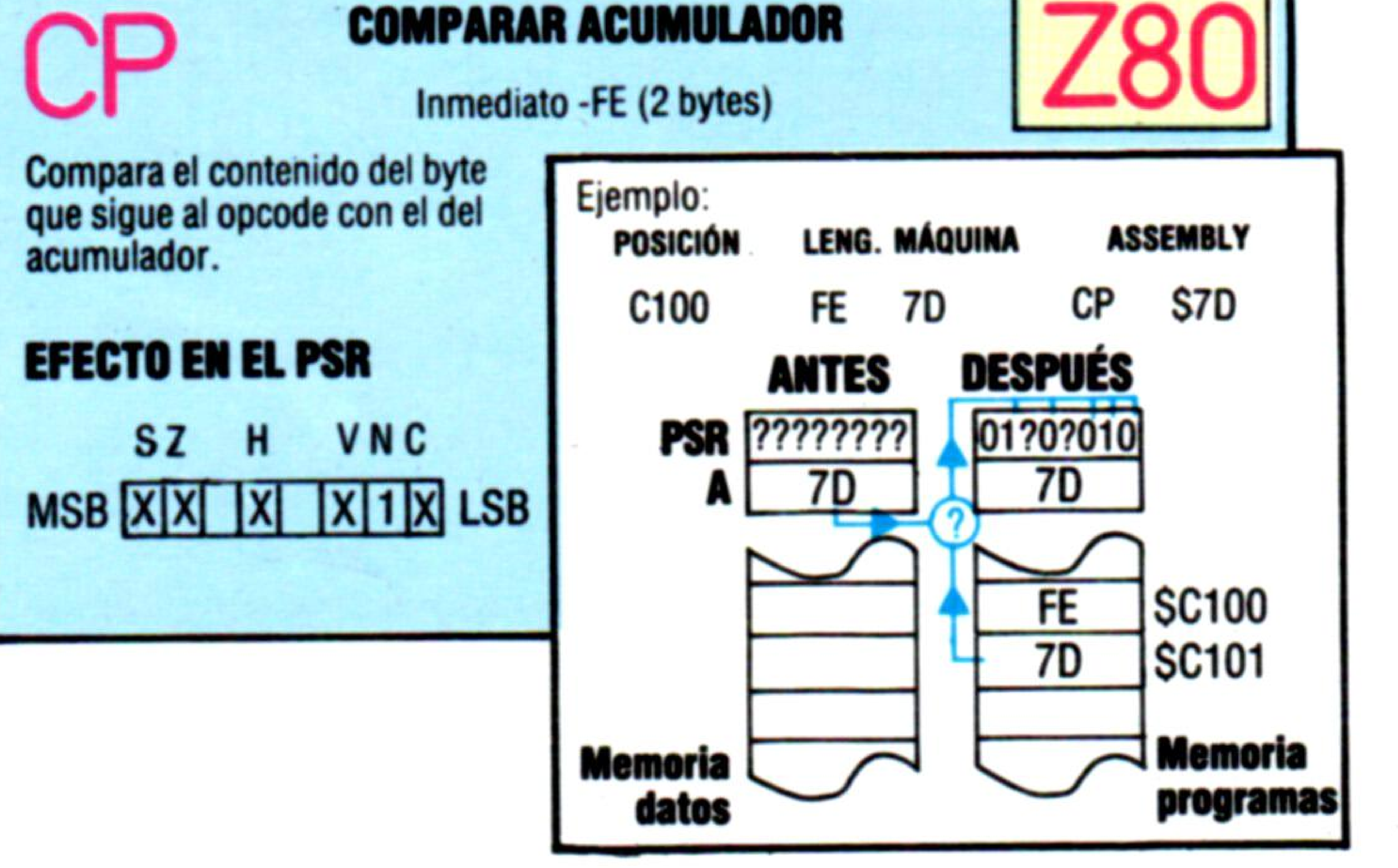

SBC
RESTAR CON ARRASTRE
Absoluto -ED (3 bytes)
6502
El contenido de la posición de memoria es restado del acumulador, y el flag de arrastre indica si hay que "quitar uno". (borrow).
EFECTO EN EL PSR
S V BD I Z C
MSB X X X X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 ED A3 59 SBC $59A3
ANTES DESPUÉS
PSR ???????1 00????01
A A7 22
$59A3 85
ED $C100
A3 $C101
59 $C102
Memoria datos
Memoria programas


SBC
RESTAR CON ARRASTRE
Inmediato -DE (2 bytes)
Z80
El contenido del byte que sigue al opcode es restado del acumulador.
EFECTO EN EL PSR
S Z H V N C
MSB X X X X X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 DE 85 SBC A,$85
ANTES DESPUÉS
PSR ???????0 00?0?000
A A7 22
DE $C100
85 $C101
Memoria datos
Memoria programas


ASL
DESPLAZ. ARITMÉTICO A IZQ.
Registro -0A (1 byte)
6502
Desplaza el contenido del byte, por medio del flag de arrastre, un bit a la izquierda, poniendo a cero el bit inferior.
EFECTO EN EL PSR
S V BD I Z C
MSB X X X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 0A ASL A
ANTES DESPUÉS
PSR ???????? 0?????11
A 10000000 00000000
0
0A $C100
Memoria datos
Memoria programas


SLA
DESPLAZ. ARITMÉTICO A IZQ.
Registro -CB (2 bytes)
Z80
Desplaza el contenido del byte, por medio del flag de arrastre, un bit a la izquierda, poniendo a cero el bit inferior.
EFECTO EN EL PSR
S Z H V N C
MSB X X 0 X 0 X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 CB 23 SLA E
ANTES DESPUÉS
PSR ???????? 01?0?001
E 10000000 00000000
0
CB $C100
23 $C101
Memoria datos
Memoria programas


CMP
COMPARAR ACUMULADOR
Absoluto -CD (3 bytes)
6502
Compara el contenido de la posición de memoria con el del acumulador.
EFECTO EN EL PSR
S V BD I Z C
MSB X X X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 CD 7E 40 CMP $407E
ANTES DESPUÉS
PSR ???????? 0?????10
A 7D 7D
?
$407E 7D
CD $C100
7E $C101
40 $C102
Memoria datos
Memoria programas


CP
COMPARAR ACUMULADOR
Inmediato -FE (2 bytes)
Z80
Compara el contenido del byte que sigue al opcode con el del acumulador.
EFECTO EN EL PSR
S Z H V N C
MSB X X X X 1 X LSB
Ejemplo:
POSICIÓN LENG. MÁQUINA ASSEMBLY
C100 FE 7D CP $7D
ANTES DESPUÉS
PSR ???????? 01?0?010
A 7D 7D
?
FE $C100
7D $C101
Memoria datos
Memoria programas

Porvenir incierto

Los problemas financieros de Dragon Data han sembrado dudas acerca del porvenir de la empresa y la comercialización de sus productos

Dragon Data se estableció originalmente como una subsidiaria de Mettoy, fabricantes de juguetes, en 1981. La intención de Mettoy no era otra que aprovecharse del *boom* de los ordenadores personales, que entonces acababa de iniciarse en Gran Bretaña. Con la ayuda financiera de la Welsh Development Agency (Agencia Galesa para el Desarrollo), se montó una fábrica en Swansea, y el Dragon 32 hizo su aparición en agosto de 1982.

La empresa optó por el microprosador 6809 de Motorola, en lugar del Z80 o del 6502, usados por la mayoría de los otros fabricantes de ordenadores personales. El sistema de circuitos del Dragon seguía el trazado recomendado por Motorola, lo que suscitó acusaciones en el sentido de que Dragon Data había basado su diseño en el ordenador Tandy Color, otro modelo que utilizaba el formato Motorola. Un efecto colateral de esto fue que los usuarios descubrieron enseguida que parte del software escrito para el CoCo se podía ejecutar también en la máquina galesa.

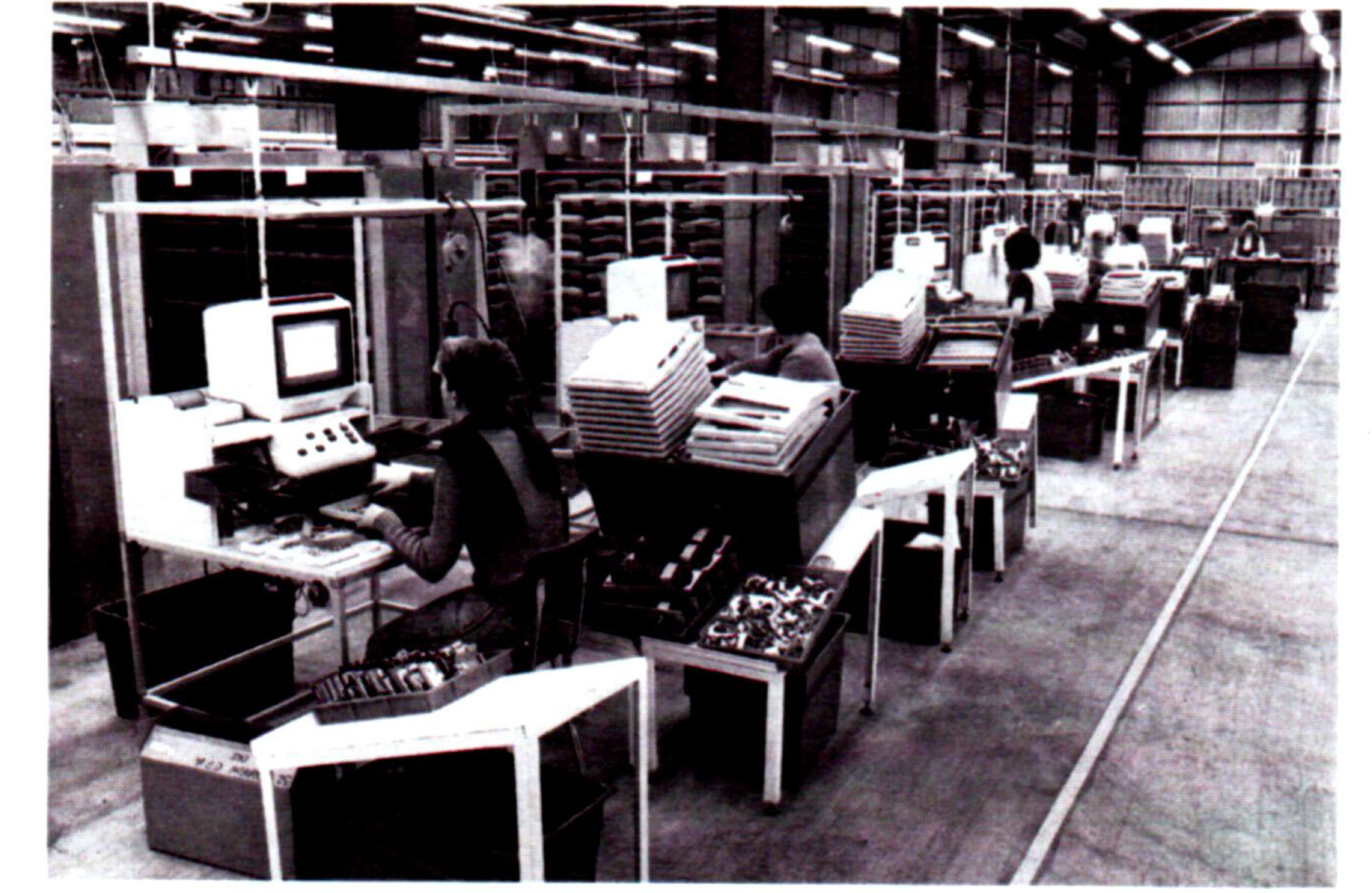
Los principales puntos para la venta del Dragon 32 fueron su BASIC Microsoft (la versión de BASIC más ampliamente utilizada) y su teclado tipo máquina de escribir de tamaño natural. En el momento en que se lanzó la máquina, en Gran Bretaña, en el sector del mercado por debajo de las 200 libras, el teclado del Dragon sólo era equiparable al del Vic-20. La estrategia de marketing de Dragon Data también desempeñó un gran papel en el éxito que obtuvo la máquina: en los meses que precedieron a la Navidad de 1982, los suministros tanto del ZX Spectrum como del BBC Micro eran escasos, y el Commodore 64 aún estaba por lanzar. Las existencias del Dragon 32 eran muy grandes y para comienzos de 1983 la empresa había vendido 32 000 máquinas. Ello se debió, en parte, a la conexión con Mettoy; las cadenas de tiendas comerciales más importantes, que siempre habían tenido grandes existencias de los juguetes de la empresa, se sentían felices al poder vender el nuevo ordenador.

Sin embargo, en el verano de 1983 Dragon Data se encontró inmersa en un serio problema financiero. La empresa estaba en plena expansión cuando Mettoy entró en bancarrota, sembrando dudas acerca del futuro de su subsidiaria galesa. Finalmente Dragon pudo salvarse gracias a un consorcio de empresas encabezado por Prutec, la rama de inversiones en alta tecnología de la gigantesca empresa de seguros Prudential. Se reunió una suma de 2,5 millones de libras y la firma contrató los servicios de un nuevo director gerente, Brian Moore, antiguo ejecutivo de GEC. Estos cambios permitieron que Dragon Data superara sus problemas de movimiento de caja, invirtiera en una nueva factoría en Port Talbot y continuara desarrollando el Dragon 64 y la unidad de disco.

El Dragon 64 tiene 64 Kbytes de RAM, un teclado perfeccionado y una interface en serie RS232C. La unidad de disco utiliza discos flexibles estándar de 5 ¼ pulgadas que operan bajo el Dragon DOS, que puede ser utilizado tanto por el modelo 32 como por el 64. Para el Dragon 64 también hay disponible una versión del eficaz sistema OS9.

Pero una sombra se cernió sobre todos los planes de Dragon en junio de 1984, cuando Prutec y la Welsch Develoment Agency se negaron a aportar más capital y la firma se vio enfrentada a profundas dificultades financieras. Había pocas perspectivas de encontrar un comprador para la empresa. En aquellos momentos ya había tres nuevas máquinas dentro de los planes para 1984, además de otros productos relacionados con los ordenadores. Uno de los micros planificados iba a responder al estándar MSX que introducían los japoneses (véase p. 621). Pero el dragón galés se ha convertido en una especie en peligro de extinción.

Nace un dragón
Dragon Data es poco habitual entre las empresas británicas de ordenadores personales por el hecho de que fabrica sus propias máquinas, cuando la mayoría de las empresas, como Sinclair y Acorn, subcontratan la producción de sus ordenadores. Recientemente Dragon ha construido una nueva fábrica en Port Talbot (West Glamorgan, en el País de Gales), que vemos en la fotografía





Expectativas frustradas
Richard Wadman, director de marketing de Dragon Data, tenía planes para vender toda una nueva gama de ordenadores Dragon, cuando la empresa se vio súbitamente en dificultades financieras

Editorial Delta, S.A.



10040
9 788485 822836